FINAL REPORT
VOLUME 2:
TASK 2: SEEKER SCENE EMULATOR

CLIN 0006

November 2, 1990

MACROSTRUCTURE LOGIC ARRAYS

Contract No. DASG60-85-C-0041
Sponsored By
The United States Army Strategic Defense Command

COMPUTER ENGINEERING RESEARCH LABORATORY
Georgia Institute of Technology
Atlanta, Georgia 30332 - 0540

Contract Data Requirements List Item F006
Period Covered: 1985-1990
Type Report: Final

DISCLAIMER

DISCLAIMER STATEMENT - The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

DISTRIBUTION CONTROL

(1) DISTRIBUTION STATEMENT - Approved for public release; distribution is unlimited.

(2) This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227 - 7013, October 1988.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT 1)Approved for public release; distribution is unlimited. 2) (continued on reverse side)
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION School of Electrical Eng. Georgia Tech 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION U.S. Army Strategic Defense Command
6c ADDRESS (City, State, and ZIP Code) Atlanta, Georgia 30332	7b ADDRESS (City, State, and ZIP Code) P.O. Box 1500 Huntsville, AL 35807-3801
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DASG60-85-C-0041
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Macrostructure Logic Arrays Volume 2, Task 2

12. PERSONAL AUTHOR(S)
C. O. Alford

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 6/28/85 TO 11/2/90	November 2, 1990	163

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Volume 2, Task 2 Seeker Scene Emulator

1. Introduction
2. Design Hardware
   - 2.1 Host
   - 2.2 Processor Array
   - 2.3 Display
   - 2.4 Interfaces
   - 2.5 Data Storage
3. Software Tools
   - 3.1 ESSING
4. Simulation Software
   - 4.1 Seeker Emulator Operation
   - 4.2 Algorithms
   - 4.3 Target/Scene Tapes
   - 4.4 Performance
5. References
6. Appendices

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Distribution statement continued
2) This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227-7013, October 1988.

FINAL REPORT

VOLUME 2

TASK 2: SEEKER SCENE EMULATOR

CLIN 0006

November 7, 1990

Authors

Andrew Henshaw, Roy Melton and Steve Gieseking

COMPUTER ENGINEERING RESEARCH LABORATORY

Georgia Institute of Technology

Atlanta, Georgia 30332 - 0540

Eugene L. Sanders	Cecil O. Alford
USASDC	Georgia Tech
Contract Monitor	Project Director

Table of Contents

1. Introduction

Under direction from the U. S. Army Strategic Defense Command, the Computer Engineering Research Laboratory at the Georgia Institute of Technology and BDM Corporation have developed a real-time Focal Plane Array Seeker Scene Emulator. This unit enhances Georgia Tech's capabilities in KEW system testing and performance demonstration.

1.1. History

As shown in Figure 1.1, the SDIO HWIL Simulation Structure (as presented by Dr. Clarence Giese) contains three paths for exercising the Signal Processing and Data Processing algorithms and hardware. Two of these methods use actual FPA hardware to generate signals for presentation to the SP and DP sub-systems. In many cases, the use of an FPA might be considered restrictive. The Georgia Tech Seeker Scene Emulator is designed to provide the third path in this simulation structure. By emulating the FPA, the Georgia Tech SSE can provide test results that would be costly or difficult to achieve using an actual FPA. The SSE can be used to fill in gaps in testing of components in stressing simulation scenarios, such as nuclear environments and high object counts.

1.2. Objectives

The FPA Seeker Scene Emulator combines advanced hardware developed at Georgia Tech with a BDM-generated database to produce signals based upon target radiometric information, seeker optical characterization, FPA detector characterization, and simulated background environments. Using real-time, positional updates, typically from the Georgia Tech Parallel Function Processor, the Seeker Scene Emulator can combine elements of the pre-computed database to form an image that is positionally and radiometrically correct.

1.3. Requirements

The Georgia Tech SSE is designed to accurately emulate FPAs with:

- up to 128 x 128 detectors
- rates up to 100 Hz
- pixel-by-pixel non-uniform response
- flexible A/D modeling, with up to 16 bits/pixel
- fully diffracted optical images
- complex environments, such as
  - nuclear redout
  - hundreds of objects
  - multiple color bands

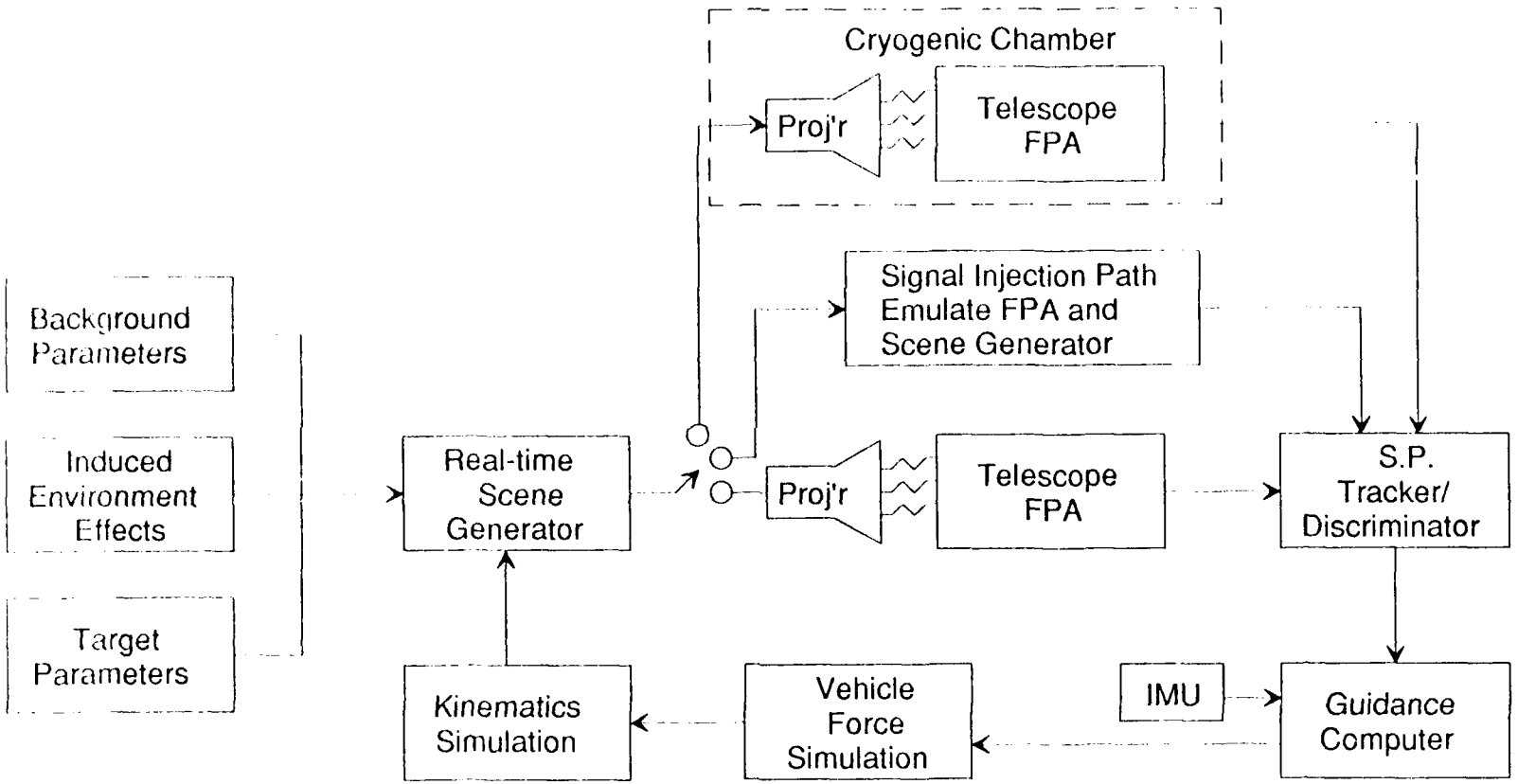

Figure 1.1: SDIO HWIL Simulation Structure

2. Design Hardware

Real-time operation of the Seeker Scene Emulator is achieved by precomputing target and noise data for a simulation, transferring this data to the Seeker Scene Emulator, and merging the target and noise data to produce a correct image. This image can then be processed as if it were data from an actual missile seeker sub-system.

Figure 2.1 shows a typical mission flight profile. For a given 6-degree-of-freedom (DOF) simulation, the variance in target range will be quite small as changes in SP and DP algorithms are evaluated. What can change significantly, however, is the frame-to-frame target line-of sight (LOS). These properties are exploited in the design of the SSE. Precomputing of the target and noise data relies upon the small variance in target range. However, the SSE must still compensate for the LOS changes in real time.

The Seeker Scene Emulator database consists of two data structures, the Noise and Target Files, which are combined in real time to form the Emulator output. The Noise File is generated at the resolution of the simulated FPA, e.g. 128 x 128 elements. The Target File, however, is generated at higher resolutions to provide sub-pixel resolution for alignment of the target image against the noise image. Currently, 16 sub-pixels of target image are generated for each pixel of noise image.

These data files are computed off-line using a 6-DOF simulation with "ideal" components (Figure 2.2). When the same simulation is run real-time on the Georgia Tech PFP with hardware emulation, LOS differences will emerge. The Seeker Scene Emulator accommodates these differences by shifting the target data with respect to the noise data. Once the shifting has occurred, the sub-pixel target data is combined with the noise data to produce an image at the established resolution of the simulated focal plane array. At this point, a simulation of the analog-to-digital conversion process transforms the image data to a specified word width and the data is streamed out of the Seeker Scene Emulator (Figure 2.3).

2.1. Host

Currently, the Host for the Seeker Emulator is a PC-AT compatible system, with a removable disk drive, a fixed disk drive, and a Transputer motherboard with module. The module consists of a T800 Transputer and 8 megabytes of memory. This processor acts as the controller for the system and the code for its operation is listed in Appendix A.

2.2. Processor Array

The FPA Seeker Scene Emulator (Figure 2.4) is a 256 processor system providing high computing performance, 512 Megabytes of memory, and flexible inter-processor communication.

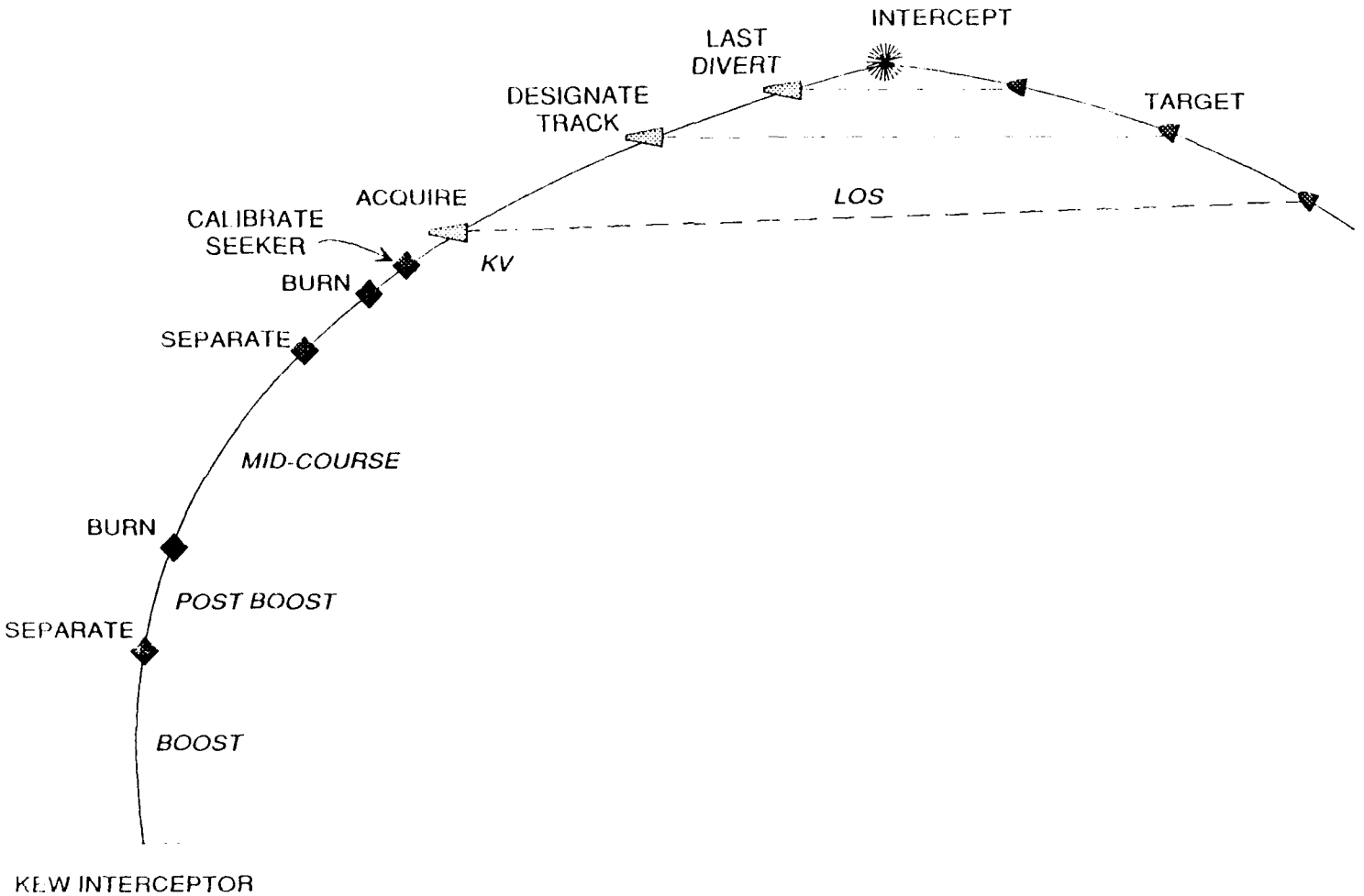

Figure 2.1: Mission Flight Profile

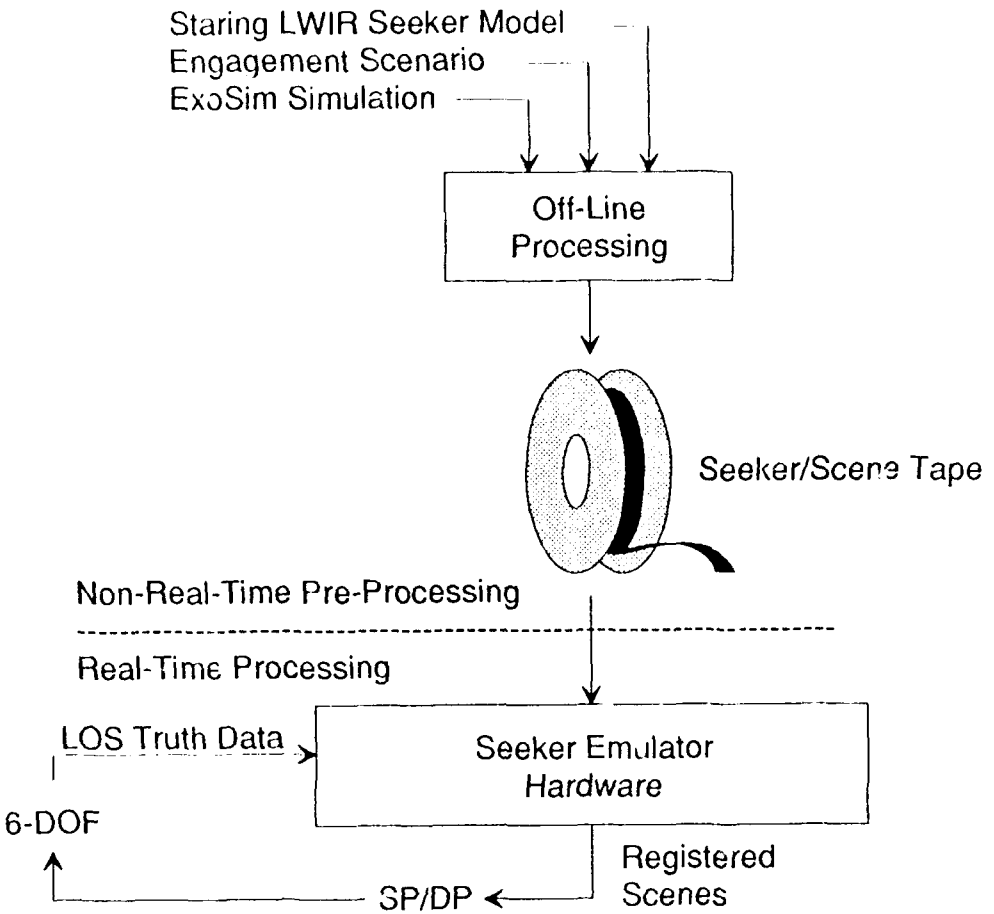

Figure 2.2: Seeker Emulator Approach

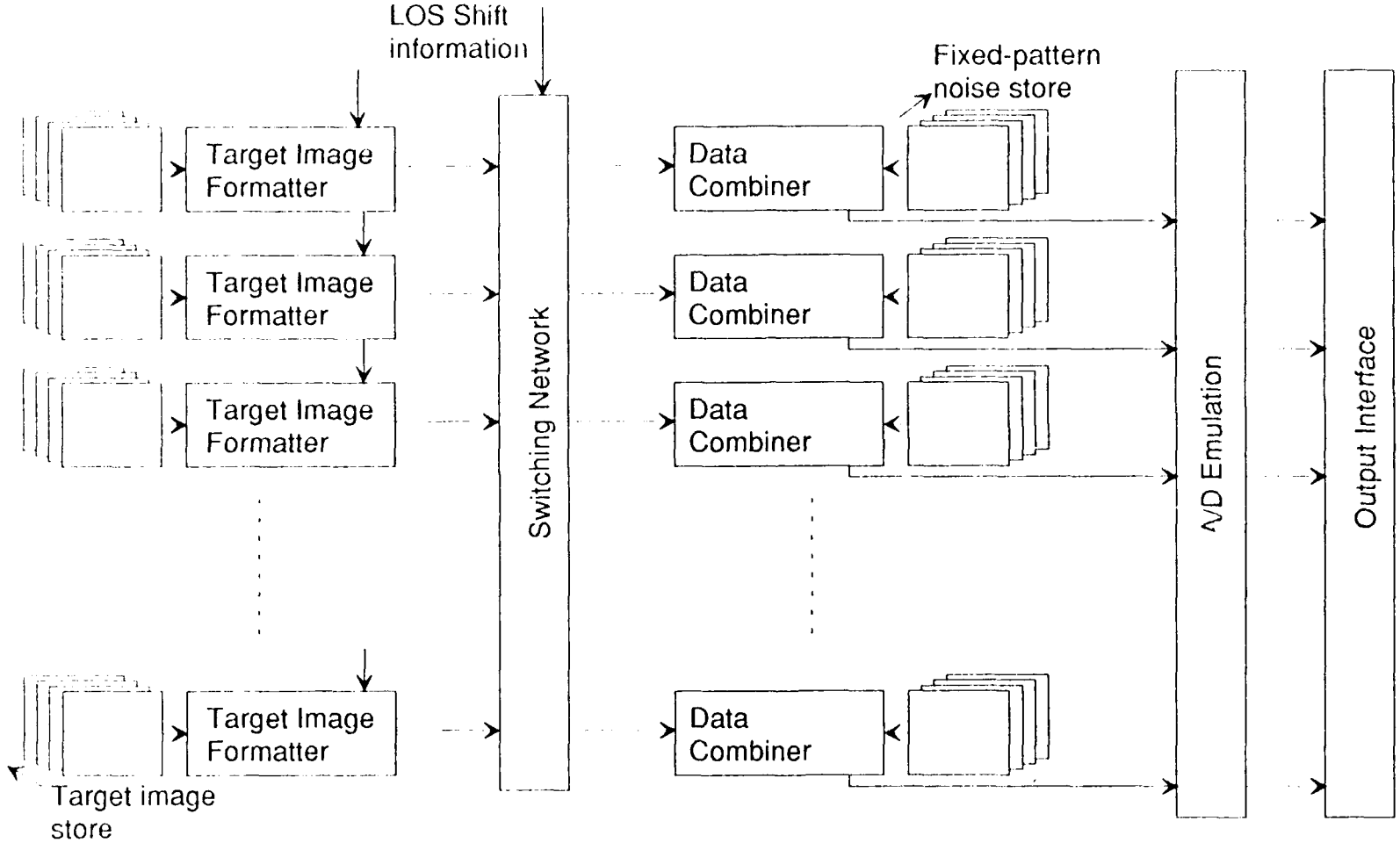

Figure 2.3: Data Construction

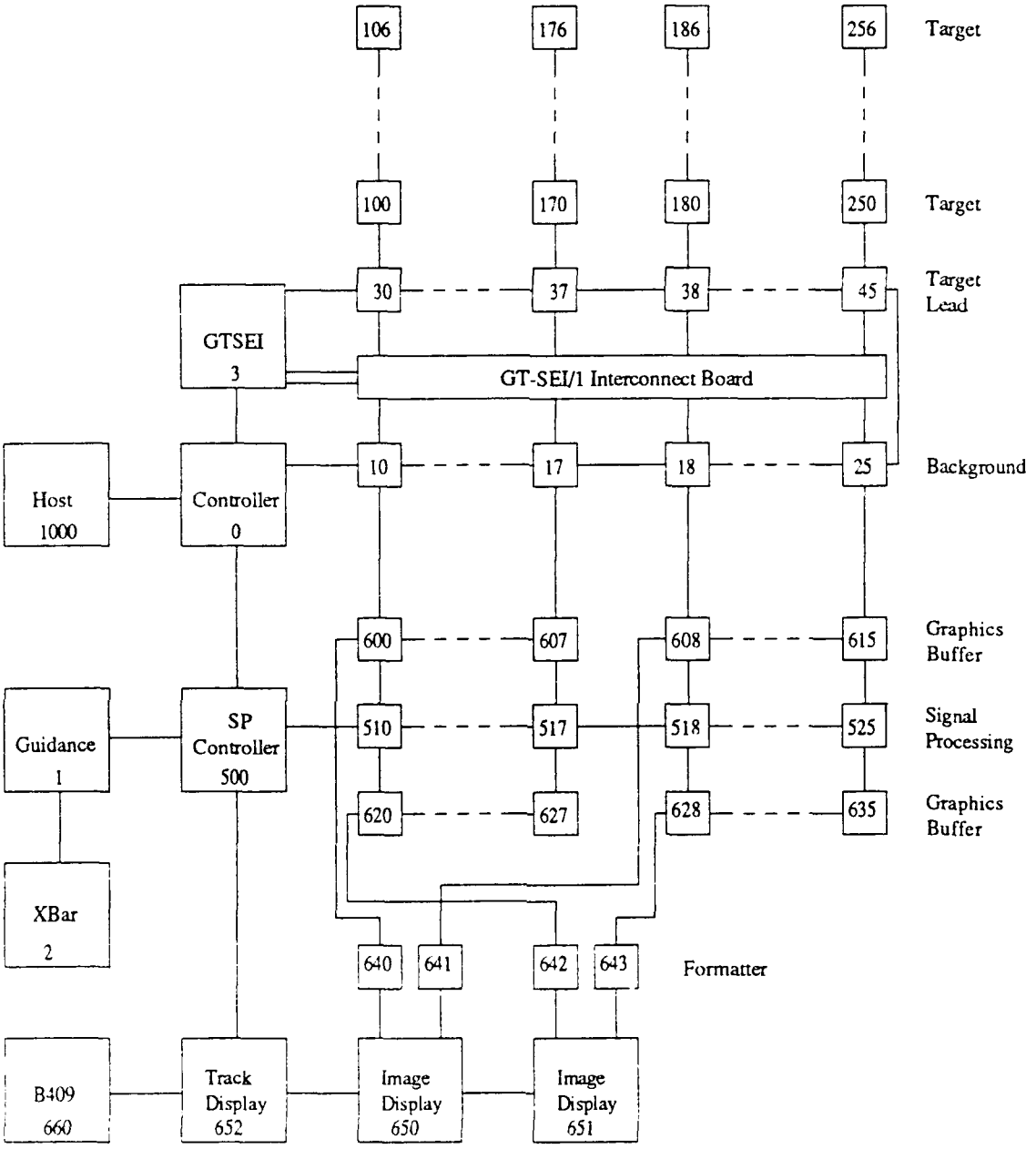

Figure 2.4: Seeker Emulator Interconnections

The current processing element is an Inmos T800 Transputer. The T800 is a 32/64 bit processor with on-chip floating-point hardware and four high-speed, serial communication channels. The processors are available on daughterboards which support 2 megabytes of dynamic RAM and 128 Kilobytes of static RAM. With the current design, real-time emulation of a 128 x 128 FPA is possible at rates over 100 frames per second.

2.3. Display

The host display uses a character-style menu that leads the user through input selections. The SSE display, on the other hand, uses a high-speed graphical interface to display a visual representation of both the raw, emulated FPA output and, optionally, processed Seeker data. The display parameters are 640 x 480 pixels and a 90 Hz refresh rate. Three processors, each with individual frame buffers, are responsible for updating the display image. One processor is allocated to each of the two FPA data arrays and a third displays range, rate, and time information. The unique design of the graphics system, based upon the Inmos B408 and B409 TRAM boards, allows all three processors to simultaneously display their results on the same monitor.

2.4. Interfaces

In addition to those interfaces which are an integral part of the Host (referenced above) and Data Storage (referenced below) sub-systems, there are two important Seeker Emulator Interfaces: the PFP interface and the Signal Processing interface.

2.4.1. PFP Interface

The PFP interface provides synchronized communication between the Seeker Emulator and the Parallel Function Processor. Using an Inmos-standard TRAM form factor, the interface provides 4 Transputer links for communication with the Seeker Emulator and a Georgia Tech-standard PFP crossbar port.

The PFP interface is mounted in the Seeker Emulator occupying a Size 2 TRAM site. Physically, the board measures approximately 3" x 2" and uses a T212 (16-bit) Transputer. The code for this processor is also given in Appendix A.

2.4.2. Signal Processing Interface

At the time of the writing of this document, the Signal Processing Interface had not been completely specified. The development of this interface is under a different contract and is mentioned here for reference purposes. It can be assumed that the interface will support variable frame rates up to 100 Hz and FPA sizes up to 128 x 128 as these are the specified capabilities of the Seeker Emulator.

2.5. Data Storage

Storage of the extremely large data files needed for operation of the Seeker Emulator is provided by an Exabyte 8mm tape drive mounted in a DEC MicroVax II Workstation. This system also houses a Transputer-interface board provided by Caplin Cybernetics.

Before running the Seeker Emulator, the data must be loaded from tape. For a 169 frame simulation, this can take over 20 minutes. The two routines 'senddata' and 'send34000' perform the actual transfer from the mass storage through the Transputer interface and to the Seeker Emulator. The 'send34000' routine is a modified version of the 'senddata' program that reads data from the tape in blocks of 34000 bytes. This is a non-standard block size, but using such a large block size greatly reduces the loading time.

3. Software Tools

Most of the software associated with the SSE is actually incorporated into the run-time program (see Appendix A). One of the important programs which is run off-line is ESSING. ESSING is the target and noise data file generator written by BDM.

3.1. ESSING (BDM manual)

Operation of the ESSING software is fully described in the ESSING USER'S MANUAL [1]. ESSING can be run on VAX-compatible systems and is available on the MicroVax II that is attached to the Seeker Scene Emulator. The data files produced can be enormous, for example, the target data file for a 169 frame simulation would occupy in excess of 175 megabytes.

4. Simulation Software

During real-time operation of the Seeker Scene Emulator, certain signal and object processing operations can optionally be emulated by the Transputer array. These implementations may not be as rigorous as those that are to be performed by custom VLSI chipsets, but they do test the standalone operation of the SSE. The algorithms that are currently implemented are non-uniformity compensation, thresholding, and hot-spot detection

4.1. Seeker Emulator Operation

The Occam source code shown in Appendix B is a single program that is executed by many processors (over 200). The assignment of processes (see Figure 2.4) is static and a relationship between the processors and procedures can be illustrated in a table.

Table 4.1: Processor Assignments

Processor Number	Procedure Name	Description
1000	Host	Menu interface for user. Loads data files. Video image storage.
0	Controller	Responsible for sending commands to all processes. Generates all timing signals.
1	Guidance	Emulates a simple guidance procedure when the PFP is not attached. Also handles communication with the XBar process.
2	XBar	This process executes on the GT-XBI module. Communicates with the PFP Crossbar.
3	GTSEI	Seeker Emulator Interconnect - this code controls the switch network. The network is re-configured for each frame depending on the actual LOS shift.
10-25	Background	Receives target data, performs non-linearization (up to 5th order) and adds noise data. Also performs simulated analog/digital conversion.

30-45	TargetLead	Shifts pointers to in-memory target data based upon actual LOS shift. Streams data out to switch network for gross shift. Passes shift information to other Lead Processors.
100-256	Target	Shifts pointers to in-memory target data based upon actual LOS shift. Streams data out to Lead Target Processors.
500	SPController	Manages messages to and from SP processes.
510-525	SignalProcessing	Performs non-uniformity compensation, thresholding, etc.
600-635	GraphicsBuffer	Reduces data flow from 8 links down to 1 link.
640-643	Formatter	Organizes data for display.
650-651	ImageDisplay	This is the process that updates the screen display.
652	TrackDisplay	Show range, time, etc. on the Seeker Display. Also show centroid of identified target.
660	B409	Process responsible for controlling the analog circuitry of the Display modules. Sets screen width, height, palette, and timing.

4.2. Algorithms

4.2.1. Non-Uniformity Compensation

The coding for the non-uniformity compensation is a complete implementation of the algorithm as defined in [2]. This process can be performed in parallel and thus can run real-time.

4.2.2. Thresholding

This version of thresholding does not implement all of the modes available to the GT VLSI Thresholding chip. Specifically, adaptive thresholding is not attempted and adjusted thresholding, while possible to do, is also not coded.

4.2.3. Hot-Spot Detection

For testing of the real-time operation, no attempt is made at clustering, centroiding, and tracking using the Transputer array. Instead a simple hot-spot detection is performed. This is sufficient if the target data files have been constructed appropriately. Obviously, for those cases when the target cannot be determined using this simplistic approach, other hardware approaches will be used.

4.3. Target/Scene Tapes

As mentioned above, the data files produced by the ESSING software are very large and strain the capacity of conventional mass storage devices. For this reason, the Seeker Scene Emulator uses an 8mm tape system (Exabyte) that can support up 4 gigabytes on a single cartridge. The tape drive is mounted in a MicroVax II, and data can be transferred to the SSE through a Transputer interface that also resides in the MicroVax.

4.4. Performance

The Georgia Tech Seeker Scene Emulator realizes all of its targeted performance goals. Emulation of a 128 x 128 FPA at frame rates of 100 Hz is possible. Additionally, the architecture is such that these capabilities could be extended if necessary.

One limitation is the generation of the off-line data files. The ESSING software takes a considerable amount of time (hours) even on high performance systems. We have investigated the parallelization of the basic ESSING software, so that this overhead could be avoided. Unfortunately, another bottleneck is the loading of the data files from tape. For a user to convert from the testing of one scenario to another would take a minimum of 30 minutes.

Georgia Tech's proposal is to leverage the hardware investment that has been made in the current SSE and to produce an Advanced Seeker Scene Emulator that would not need to have voluminous data files generated off-line. Testing of multiple scenarios could then proceed at a rapid pace. Working with BDM Corp. and Teledyne Brown Engineering, we feel that we have a reasonable approach to a Seeker Scene Emulator that could surpass the functionality of the hardware-software described in this document.

5. References

[1] BDM Corporation, *ESSING User's Manual*, BDM/HTV-89-0796-TR, U. S. Army Strategic Defense Command, Contract No.: DASG60-87-C-0111, 5 December 1989.

[2] Georgia Institute of Technology, Computer Engineering Research Laboratory, *Signal Processing Algorithms - Georgia Tech Benchmark*, U. S. Army Strategic Defense

Command, Contract No.: DASG60-85-C-0041, Special Technical Report No. STR-0142-90-008, 27 February 1990.

6. Appendices

6.1. Appendix A: Seeker Scene Emulator Publications

BDM Corporation, *Signal Processing Concepts and Algorithms*, BDM/HTV-89-0002-BR, U. S. Army Strategic Defense Command, Contract No.: DASG60-85-C-0041, 12 January 1989.

BDM Corporation, *ESSING User's Manual*, BDM/HTV-89-0796-TR, U. S. Army Strategic Defense Command, Contract No.: DASG60-87-C-0111, 5 December 1989.

BDM Corporation, *EXOSEEK Version 1.0: A Simulation of the LATS Seeker*, U. S. Army Strategic Defense Command, Contract No.: DASG60-87-C-0111, 31 January 1990.

BDM Corporation, *EXOSEEK Version 2.0: A Simulation of the LATS Seeker*, U. S. Army Strategic Defense Command, Contract No: DASG60_87-C-0111, 15 May 1990.

Georgia Institute of Technology, Computer Engineering Research Laboratory, *Macrostructure Logic Arrays Volumes 1,-3*, U. S. Army Strategic Defense Command, Contract No.: DASG60-85-C-0041, 20 July 1989.

Georgia Institute of Technology, Computer Engineering Research Laboratory, *Signal Processing Algorithms - Georgia Tech Benchmark*, U. S. Army Strategic Defense Command, Contract No.: DASG60-85-C-0041, Special Technical Report No. STR-0142-90-008, 27 February 1990.

Teledyne Brown Engineering, *GN&C Lab Seeker Emulator - An Assessment and Recommendations*, U. S. Army Strategic Defense Command, Contract No.: DASG60-87-C-0042, May 1988.

6.2. Appendix B: Seeker Scene Emulator Operation Programs

6.2.1. Occam Source

6.2.1.1.Seeker Program File "seeker.pgm"

```
--{{{  SC HostSeek
--:::A  4 10
#USE "HostSeek.c8h"
--{{{F HostSeek
--:::F hostseek.OCC
--}}}
--}}}
--{{{  SC Controller
--:::A  4 10
#USE "controll.t8h"
--{{{F Controller
--:::F CONTROLL.OCC
--}}}
--}}}
--{{{  SC Guidance
--:::A  4 10
#USE "guidance.t8h"
--{{{F Guidance
```

```
--:::F GUIDANCE.OCC
--}}}
--}}}
--{{{  SC XBar
--:::A  4 10
#USE "xbar.t2h"
--{{{F XBar
--:::F XBAR.OCC
--}}}
--}}}
--{{{  SeekerEmulator
#USE "gtsei.t2h"
--{{{  SC GTSEI
--:::A  4 10
--{{{F gtsei.occ
--:::F GTSEI.OCC
--}}}
--}}}

#USE "backgrou.t8h"
--{{{  SC Background
--:::A  4 10
--{{{F Background
--:::F BACKGROU.OCC
--}}}
--}}}

#USE "targetle.t8h"
--{{{  SC TargetLead
--:::A  4 10
--{{{F TargetLead
--:::F TARGETLE.OCC
--}}}
--}}}

#USE "target.t8h"
--{{{  SC Target
--:::A  4 10
--{{{F Target
--:::F TARGET.OCC
--}}}
--}}}
--}}}
--{{{  SignalProcessing
#USE "spcontro.t8h"
--{{{  SC SPController
--:::A  4 10
--{{{F SPController
--:::F SPCONTRO.OCC
--}}}
--}}}
#USE "sp.t4h"
--{{{  SC SP
--:::A  4 10
--{{{F SP
--:::F SP.OCC
--}}}
--}}}
--}}}
--{{{  Graphics
#USE "firstbuf.t8h"
--{{{  SC FirstBuffer
--:::A  4 10
--{{{F FirstBuffer
```

```
--:::F FIRSTBUF.OCC
--}}}
--}}}

#USE "secondbu.t8h"
--{{{  SC SecondBuffer
--:::A  4 10
--{{{F SecondBuffer
--:::F SECONDBU.OCC
--}}}
--}}}

#USE "formatte.t8h"
--{{{  SC Formatter
--:::A  4 10
--{{{F Formatter
--:::F FORMATTE.OCC
--}}}
--}}}

#USE "imagedis.c8h"
--{{{  SC ImageDisplay
--:::A  4 10
--{{{F ImageDisplay
--:::F IMAGEDIS.OCC
--}}}
--}}}

#USE "trackdis.c8h"
--{{{  SC TrackDisplay
--:::A  4 10
--{{{F TrackDisplay
--:::F TRACKDIS.OCC
--}}}
--}}}

#USE "b409stub.c2h"
--{{{  SC B409.stub
--:::A  4 10
--{{{F B409stub
--:::F B409stub.OCC
--}}}
--}}}
--}}}
--{{{  configuration
--{{{  constants
VAL  image.shift  IS  9 :
--}}}
--{{{  link definitions
VAL  link0out   IS  0 :
VAL  link1out   IS  1 :
VAL  link2out   IS  2 :
VAL  link3out   IS  3 :
VAL  link0in    IS  4 :
VAL  link1in    IS  5 :
VAL  link2in    IS  6 :
VAL  link3in    IS  7 :
--}}}
--{{{  channels
CHAN OF ANY   Controller.Host,        Host.Controller       :
CHAN OF ANY   Controller.GTSEI,       GTSEI.Controller      :
CHAN OF ANY   Controller.SPC,         SPC.Controller        :
CHAN OF ANY   Crossbar0,              Crossbar1             :
CHAN OF ANY   Guidance.XBar,          XBar.Guidance         :
```

```
CHAN OF ANY   Guidance.SPController,  SPController.Guidance :
CHAN OF ANY   SPController.Graphics,  Graphics.SPController :

[16][8]  CHAN OF ANY  Target.up,       Target.down  :
[16]     CHAN OF ANY  Target.forward,  Target.back  :
[16]     CHAN OF ANY  Target.GTSEI,    GTSEI.BG     :
[17]     CHAN OF ANY  BG.forward,      BG.back      :
[16]     CHAN OF ANY  BG.SP                         :
[16]     CHAN OF ANY  Graphics.SP,     SP.Graphics  :
[17]     CHAN OF ANY  SP.forward,      SP.back      :

[9]      CHAN OF ANY  Extract0, Extract1, Extract2, Extract3 :
[4]      CHAN OF ANY  Format.Graphics :
[4]      CHAN OF ANY  Graphics.forward, Graphics.back :
--}}}
PLACED PAR
  --{{{  HostSeeker
  PROCESSOR 1000 T8
    PLACE Host.Controller   AT link2out :
    PLACE Controller.Host   AT link2in :
    HostSeeker (Controller.Host, Host.Controller)
  --}}}
  --{{{  Controller
  PROCESSOR  0  T8

    PLACE  Host.Controller      AT  link1in  :
    PLACE  GTSEI.Controller     AT  link3in  :
    PLACE  BG.back[0]           AT  link0in  :
    PLACE  SPC.Controller       AT  link2in  :
    PLACE  Controller.Host      AT  link1out :
    PLACE  Controller.GTSEI     AT  link3out :
    PLACE  BG.forward[0]        AT  link0out :
    PLACE  Controller.SPC       AT  link2out :

    Controller( Host.Controller,       Controller.Host,
                GTSEI.Controller,      Controller.GTSEI,
                BG.back[0],            BG.forward[0],
                SPC.Controller,        Controller.SPC  )
  --}}}
  --{{{  Guidance
  PROCESSOR  1  T8

    PLACE  SPController.Guidance     AT  link1in  :
    PLACE  XBar.Guidance             AT  link0in  :
    PLACE  Guidance.SPController     AT  link1out :
    PLACE  Guidance.XBar             AT  link0out :

    Guidance( SPController.Guidance, Guidance.SPController,
              XBar.Guidance,         Guidance.XBar          )
  --}}}
  --{{{  XBar
  PROCESSOR  2  T2

    PLACE  Guidance.XBar   AT  link1in  :
    PLACE  XBar.Guidance   AT  link1out :

    XBar( Guidance.XBar,  XBar.Guidance )
  --}}}
  --{{{  SeekerEmulator
  --{{{  GTSEI
  PROCESSOR  3  T2

    PLACE  Controller.GTSEI     AT  link1in  :
    PLACE  Target.back[0]       AT  link2in  :
```

```
    PLACE  GTSEI.Controller      AT  link1out :
    PLACE  Target.forward[0]     AT  link2out :
    PLACE  Crossbar0             AT  link0out :
    PLACE  Crossbar1             AT  link3out :

    GTSEI( Controller.GTSEI,   GTSEI.Controller,
           Target.back[0],     Target.forward[0],
           Crossbar0,          Crossbar1          )
  --}}}
  --{{{  Background
  PLACED PAR i = 0 FOR 16

    PROCESSOR  10+i  T8

      PLACE  GTSEI.BG[i]         AT  link3in  :
      PLACE  BG.forward[i]       AT  link1in  :
      PLACE  BG.back[i+1]        AT  link2in  :
      PLACE  BG.SP[i]            AT  link0out :
      PLACE  BG.back[i]          AT  link1out :
      PLACE  BG.forward[i+1]     AT  link2out :

      Background( GTSEI.BG[i],       BG.SP[i],
                  BG.forward[i],     BG.back[i],
                  BG.back[i+1],      BG.forward[i+1], i )
  --}}}
  --{{{  TargetLead
  PLACED PAR i = 0 FOR 15

    PROCESSOR  30+i  T8

      PLACE  Target.down[i][0]       AT  link2in  :
      PLACE  Target.forward[i]       AT  link0in  :
      PLACE  Target.back[i+1]        AT  link3in  :
      PLACE  Target.GTSEI[i]         AT  link1out :
      PLACE  Target.up[i][0]         AT  link2out :
      PLACE  Target.back[i]          AT  link0out :
      PLACE  Target.forward[i+1]     AT  link3out :

           TargetLead(  Target.down[i][0],              Target.up[i][0],
Target.GTSEI[i],
                  Target.forward[i],  Target.back[i],
                     Target.back[i+1],         Target.forward[i+1],    i
)

  PROCESSOR  45  T8
    VAL i IS 15 :
    PLACE  Target.down[i][0]       AT  link2in  :
    PLACE  Target.forward[i]       AT  link0in  :
    PLACE  BG.forward[16]          AT  link3in  :
    PLACE  Target.GTSEI[i]         AT  link1out :
    PLACE  Target.up[i][0]         AT  link2out :
    PLACE  Target.back[i]          AT  link0out :
    PLACE  BG.back[16]             AT  link3out :

    TargetLead( Target.down[i][0],  Target.up[i][0],   Target.GTSEI[i],
                Target.forward[i],  Target.back[i],
                   BG.forward[16],                    BG.back[16],     i
)
  --}}}
  --{{{  Target
  PLACED PAR i= 0 FOR 15
    PLACED PAR j = 1 FOR 7

      PROCESSOR  100+((i*10)+j)  T8
```

```
      PLACE  Target.up[i][j-1]    AT  link1in  :
      PLACE  Target.down[i][j]    AT  link2in  :
      PLACE  Target.down[i][j-1]  AT  link1out :
      PLACE  Target.up[i][j]      AT  link2out :

      Target( Target.up[i][j-1],   Target.down[i][j-1],
              Target.down[i][j],   Target.up[i][j], i, j )
--}}}
--}}}
--{{{  SignalProcessing
--{{{  SPController
PROCESSOR  500  T8

  PLACE  Controller.SPC           AT  link1in  :
  PLACE  Guidance.SPController    AT  link2in  :
  PLACE  Graphics.SPController    AT  link3in  :
  PLACE  SP.back[0]               AT  link0in  :
  PLACE  SPC.Controller           AT  link1out :
  PLACE  SPController.Guidance    AT  link2out :
  PLACE  SPController.Graphics    AT  link3out :
  PLACE  SP.forward[0]            AT  link0out :

  SPController( Controller.SPC,          SPC.Controller,
                Guidance.SPController,   SPController.Guidance,
                Graphics.SPController,   SPController.Graphics,
                SP.back[0],              SP.forward[0]          )
--}}}
--{{{  SP
PLACED PAR i = 0 FOR 16

  PROCESSOR  510+i  T4

    PLACE  Graphics.SP[i]       AT  link3in  :
    PLACE  SP.forward[i]        AT  link1in  :
    PLACE  SP.back[i+1]         AT  link2in  :
    PLACE  SP.Graphics[i]       AT  link0out :
    PLACE  SP.back[i]           AT  link1out :
    PLACE  SP.forward[i+1]      AT  link2out :

    SP( Graphics.SP[i],      SP.Graphics[i],
        SP.forward[i],       SP.back[i],
        SP.back[i+1],        SP.forward[i+1], i )
--}}}
--}}}
--{{{  Graphics
--{{{  FirstBuffer
PLACED PAR i = 0 FOR 8

  PROCESSOR  600+i  T8

    PLACE  BG.SP[i]          AT  link3in  :
    PLACE  Extract0[i+1]     AT  link2in  :
    PLACE  Graphics.SP[i]    AT  link0out :
    PLACE  Extract0[i]       AT  link1out :

    FirstBuffer( BG.SP[i],         Graphics.SP[i],
                 Extract0[i+1],    Extract0[i],     i, image.shift )

PLACED PAR i = 0 FOR 8

  PROCESSOR  608+i  T8

    PLACE  BG.SP[i+8]        AT  link3in  :
```

```
    PLACE  Extract1[i+1]     AT  link1in  :
    PLACE  Graphics.SP[i+8] AT  link0out :
    PLACE  Extract1[i]       AT  link2out :

    FirstBuffer( BG.SP[i+8],       Graphics.SP[i+8],
                 Extract1[i+1],    Extract1[i],       i, image.shift )
--}}}
--{{{  SecondBuffer
PLACED PAR i = 0 FOR 8

  PROCESSOR  620+i  T8

    PLACE  SP.Graphics[i]    AT  link3in  :
    PLACE  Extract2[i+1]     AT  link2in  :
    PLACE  Extract2[i]       AT  link1out :

    SecondBuffer( SP.Graphics[i],
                  Extract2[i+1],   Extract2[i], i, image.shift )

PLACED PAR i = 0 FOR 8

  PROCESSOR  628+i  T8

    PLACE  SP.Graphics[i+8]  AT  link3in  :
    PLACE  Extract3[i+1]     AT  link1in  :
    PLACE  Extract3[i]       AT  link2out :

    SecondBuffer( SP.Graphics[i+8],
                  Extract3[i+1],   Extract3[i], i, image.shift )
--}}}
--{{{  Formatters
PROCESSOR  640  T8

  PLACE  Extract0[0]        AT  link0in  :
  PLACE  Format.Graphics[0] AT  link1out :

  Formatter( Extract0[0],  Format.Graphics[0] )


PROCESSOR  641  T8

  PLACE  Extract1[0]        AT  link0in  :
  PLACE  Format.Graphics[1] AT  link1out :

  Formatter( Extract1[0],  Format.Graphics[1] )


PROCESSOR  642  T8

  PLACE  Extract2[0]        AT  link0in  :
  PLACE  Format.Graphics[2] AT  link1out :

  Formatter( Extract2[0],  Format.Graphics[2] )


PROCESSOR  643  T8

  PLACE  Extract3[0]        AT  link0in  :
  PLACE  Format.Graphics[3] AT  link1out :

  Formatter( Extract3[0],  Format.Graphics[3] )
--}}}
--{{{  ImageDisplays
PROCESSOR  650  T8
```

```
    CHAN OF ANY temp1, temp2 :
    PLACE  Format.Graphics[0]     AT  link1in  :
    PLACE  Format.Graphics[1]     AT  link2in  :
    PLACE  Graphics.back[1]       AT  link3in  :
    PLACE  Graphics.forward[2]    AT  link0in  :
    PLACE  Graphics.forward[1]    AT  link3out :
    PLACE  Graphics.back[2]       AT  link0out :

    ImageDisplay( Format.Graphics[0],   Format.Graphics[1],
                  Graphics.back[1], Graphics.forward[1],
                  Graphics.forward[2],  Graphics.back[2], 0,  1000   )

  PROCESSOR  651  T8

    PLACE  Format.Graphics[2]     AT  link2in  :
    PLACE  Format.Graphics[3]     AT  link1in  :
    PLACE  Graphics.back[2]       AT  link3in  :
    PLACE  Graphics.forward[3]    AT  link0in  :
    PLACE  Graphics.forward[2]    AT  link3out :
    PLACE  Graphics.back[3]       AT  link0out :

    ImageDisplay( Format.Graphics[2],   Format.Graphics[3],
                  Graphics.back[2],     Graphics.forward[2],
                  Graphics.forward[3],  Graphics.back[3], 1,  1000   )
  --}}}
  --{{{  TrackDisplay
  PROCESSOR  652  T8

    PLACE  SPController.Graphics   AT  link1in  :
    PLACE  Graphics.back[0]        AT  link3in  :
    PLACE  Graphics.forward[1]     AT  link0in  :
    PLACE  Graphics.SPController   AT  link1out :
    PLACE  Graphics.forward[0]     AT  link3out :
    PLACE  Graphics.back[1]        AT  link0out :

    TrackDisplay( SPController.Graphics,  Graphics.SPController,
                  Graphics.back[0],       Graphics.forward[0],
                  Graphics.forward[1],    Graphics.back[1]       )
  --}}}
  --{{{  B409
  PROCESSOR  660  T2

    PLACE  Graphics.forward[0]  AT  link0in  :
    PLACE  Graphics.back[0]     AT  link0out :

    B409.stub( Graphics.forward[0], Graphics.back[0] )
  --}}}
  --}}}
--}}}
```

6.2.1.2. PROC B409 "b409.occ"

```
--{{{  SC B409
--:::A  3 10
--{{{  B409
#INCLUDE "crtc.inc"
PROC B409 ( CHAN OF CRTC command )

  --{{{  defs
  VAL bpw.shift IS 1:
  VAL mint IS #8000:
  --}}}
  --{{{  pointers to hardware registers
```

```
  VAL ChannelModeSelect.p      IS #B000:
  --{{{  ChannelA defs
  VAL ChannelAPixelAddressW.p IS #0000:
  VAL ChannelAColorValue.p     IS #0400:
  VAL ChannelAPixelMask.p      IS #0800:
  VAL ChannelAPixelAddressR.p IS #0C00:
  --}}}
  --{{{  ChannelB defs
  VAL ChannelBPixelAddressW.p IS #1000:
  VAL ChannelBColorValue.p     IS #1400:
  VAL ChannelBPixelMask.p      IS #1800:
  VAL ChannelBPixelAddressR.p IS #1C00:
  --}}}
  --{{{  ChannelC defs
  VAL ChannelCPixelAddressW.p IS #2000:
  VAL ChannelCColorValue.p     IS #2400:
  VAL ChannelCPixelMask.p      IS #2800:
  VAL ChannelCPixelAddressR.p IS #2C00:
  --}}}
  VAL ParameterFIFO.p          IS #A000:
  VAL StatusRegister.p         IS #A000:
  VAL CommandFIFO.p            IS #A002:
  VAL FIFORead.p               IS #A002:
  --}}}
  --{{{  hardware placements
  INT ChannelModeSelect      :
  --{{{  ChannelA declarations
  INT ChannelAPixelAddressW :
  INT ChannelAColorValue     :
  INT ChannelAPixelMask      :
  INT ChannelAPixelAddressR :
  --}}}
  --{{{  ChannelB declarations
  INT ChannelBPixelAddressW :
  INT ChannelBColorValue     :
  INT ChannelBPixelMask      :
  INT ChannelBPixelAddressR :
  --}}}
  --{{{  ChannelC declarations
  INT ChannelCPixelAddressW :
  INT ChannelCColorValue     :
  INT ChannelCPixelMask      :
  INT ChannelCPixelAddressR :
  --}}}
  INT ParameterFIFO          :
  INT StatusRegister         :
  INT CommandFIFO            :
  INT FIFORead               :
  --{{{  ChannelA placements
  PLACE ChannelAPixelAddressW AT (ChannelAPixelAddressW.p >< mint) >>
bpw.shift :
  PLACE ChannelAColorValue    AT (ChannelAColorValue.p    >< mint) >>
bpw.shift :
  PLACE ChannelAPixelMask     AT (ChannelAPixelMask.p     >< mint) >>
bpw.shift :
  PLACE ChannelAPixelAddressR AT (ChannelAPixelAddressR.p >< mint) >>
bpw.shift :
  --}}}
  --{{{  ChannelB placements
  PLACE ChannelBPixelAddressW AT (ChannelBPixelAddressW.p >< mint) >>
bpw.shift :
  PLACE ChannelBColorValue    AT (ChannelBColorValue.p    >< mint) >>
bpw.shift :
```

```
  PLACE ChannelBPixelMask       AT (ChannelBPixelMask.p       >< mint) >>
bpw.shift :
  PLACE ChannelBPixelAddressR   AT (ChannelBPixelAddressR.p   >< mint) >>
bpw.shift :
  --}}}
  --{{{  ChannelC placements
  PLACE ChannelCPixelAddressW   AT (ChannelCPixelAddressW.p   >< mint) >>
bpw.shift :
  PLACE ChannelCColorValue      AT (ChannelCColorValue.p      >< mint) >>
bpw.shift :
  PLACE ChannelCPixelMask       AT (ChannelCPixelMask.p       >< mint) >>
bpw.shift :
  PLACE ChannelCPixelAddressR   AT (ChannelCPixelAddressR.p   >< mint) >>
bpw.shift :
  --}}}
  PLACE ChannelModeSelect       AT (ChannelModeSelect.p       >< mint) >>
bpw.shift :
  PLACE ParameterFIFO           AT (ParameterFIFO.p           >< mint) >>
bpw.shift :
  PLACE StatusRegister          AT (StatusRegister.p          >< mint) >>
bpw.shift :
  PLACE CommandFIFO             AT (CommandFIFO.p             >< mint) >>
bpw.shift :
  PLACE FIFORead                AT (FIFORead.p                >< mint) >>
bpw.shift :
  --}}}
  --{{{  CRTC commands
  VAL CTRCReset IS #00:
  VAL CTRCBctrl IS #0D:
  --}}}
  --{{{  set.colour
  PROC set.colour ( VAL INT channel, colour, red, green, blue )
    -- set up a colour in the G170 colour look up table
    CASE channel
      INT channel.A
        --{{{
        SEQ
          ChannelAPixelAddressW := 255  - (colour /\ #FF)
          ChannelAColorValue := red
          ChannelAColorValue:= green
          ChannelAColorValue := blue
        --}}}
      INT channel.B
        --{{{
        SEQ
          ChannelBPixelAddressW := 255  - (colour /\ #FF)
          ChannelBColorValue := red
          ChannelBColorValue:= green
          ChannelBColorValue := blue
        --}}}
      INT channel.C
        --{{{
        SEQ
          ChannelCPixelAddressW := 255  - (colour /\ #FF)
          ChannelCColorValue := red
          ChannelCColorValue:= green
          ChannelCColorValue := blue
        --}}}
  :
  --}}}
  --{{{  writeCRTC
  PROC writeCRTC(INT address, VAL INT data)
    TIMER time:
    INT now:
```

```
    SEQ
      address := data
      time ? now
      time ? AFTER (now PLUS 2)
:
--}}}
--{{{  init.G170
PROC init.G170 (VAL INT channel, table)
  INT red, green, blue:
  SEQ
    ChannelAPixelMask := #FF
    IF
      --{{{  table 0
      table = 0
        VAL scale IS [0, 18, 36, 54] :
        VAL bias IS 9 :
        SEQ
          set.colour (channel, 0, 0, 0, 0)       -- black
          SEQ i = 1 FOR 255
            INT ix :
            SEQ
              blue := scale[(i>>4)/\3]
              green := scale[(i>>2)/\3]
              red := scale[i/\3]
              IF
                i >= #C0
                  blue := blue + bias
                i >= #80
                  green := green + bias
                i >= #40
                  red := red + bias
                TRUE
                  SKIP
              CASE channel
                --{{{  channel.A
                INT channel.A
                  SEQ
                    ChannelAColorValue := red
                    ChannelAColorValue := green
                    ChannelAColorValue := blue
                --}}}
                --{{{  channel.B
                INT channel.B
                  SEQ
                    ChannelBColorValue := red
                    ChannelBColorValue := green
                    ChannelBColorValue := blue
                --}}}
                --{{{  channel.C
                INT channel.C
                  SEQ
                    ChannelCColorValue := red
                    ChannelCColorValue := green
                    ChannelCColorValue := blue
                --}}}
      --}}}
      --{{{  COMMENT table 1
      --:::A  0 0
      --{{{  table 1
      table = 1
        SEQ
          --{{{    0  -  15 grey scale
          red := -1
          green := -1
```

```
blue := -1
SEQ i = 0 FOR 16
  SEQ
    red := red + 4
    green := green + 4
    blue := blue + 4
    set.colour (channel, i, red, green, blue)
--}}}
--{{{   16  -   32 red scale
red := -1
green := 0
blue := 0
SEQ i = 16 FOR 16
  SEQ
    red := red + 4
    set.colour (channel, i, red, green, blue)
--}}}
--{{{   32  -   47 green scale
red := 0
green := -1
blue := 0
SEQ i = 32 FOR 16
  SEQ
    green := green + 4
    set.colour (channel, i, red, green, blue)
--}}}
--{{{   48  -   63 blue scale
red := 0
green := 0
blue := -1
SEQ i = 48 FOR 16
  SEQ
    blue := blue + 4
    set.colour (channel, i, red, green, blue)
--}}}
--{{{   64  -   79 yellow scale
red := -1
green := -1
blue := 0
SEQ i = 64 FOR 16
  SEQ
    red := red + 4
    green := green + 4
    set.colour (channel, i, red, green, blue)
--}}}
--{{{   80  -   95 cyan scale
red := 0
green := -1
blue := -1
SEQ i = 80 FOR 16
  SEQ
    green := green + 4
    blue := blue + 4
    set.colour (channel, i, red, green, blue)
--}}}
--{{{   96  -  111 magenta scale
red := -1
green := 0
blue := -1
SEQ i = 96 FOR 16
  SEQ
    red := red + 4
    blue := blue + 4
    set.colour (channel, i, red, green, blue)
```

```
--}}}
--{{{  112  -  127 red & green scale with third blue
red := 63
green := -1
blue := 21
SEQ i = 112 FOR 16
  SEQ
    green := green + 4
    set.colour (channel, i, red, green, blue)
    red := red - 4
--}}}
--{{{  128  -  143 green & blue scale with third red
red := 21
green := 63
blue := -1
SEQ i = 128 FOR 16
  SEQ
    blue := blue + 4
    set.colour (channel, i, red, green, blue)
    green := green - 4
--}}}
--{{{  144  -  159 blue & redscale with third green
red := -1
green := 21
blue := 63
SEQ i = 144 FOR 16
  SEQ
    red := red + 4
    set.colour (channel, i, red, green, blue)
    blue := blue - 4
--}}}
--{{{  160  -  175 red & green scale with two-thirds blue
red := 63
green := -1
blue := 42
SEQ i = 160 FOR 16
  SEQ
    green := green + 4
    set.colour (channel, i, red, green, blue)
    red := red - 4
--}}}
--{{{  176  -  191 green & blue scale with two-thirds red
red := 42
green := 63
blue := -1
SEQ i = 176 FOR 16
  SEQ
    blue := blue + 4
    set.colour (channel, i, red, green, blue)
    green := green - 4
--}}}
--{{{  192  -  207 blue & red scale with two-thirds green
red := -1
green := 42
blue := 63
SEQ i = 192 FOR 16
  SEQ
    red := red + 4
    set.colour (channel, i, red, green, blue)
    blue := blue - 4
--}}}
--{{{  208  -  223 red & green scale with full blue
red := 63
green := -1
```

```
            blue := 63
            SEQ i = 208 FOR 16
              SEQ
                green := green + 4
                set.colour (channel, i, red, green, blue)
                red := red - 4
            --}}}
            --{{{  224  -  239 green & blue scale with full red
            red := 63
            green := 63
            blue := -1
            SEQ i = 224 FOR 16
              SEQ
                blue := blue + 4
                set.colour (channel, i, red, green, blue)
                green := green - 4
            --}}}
            --{{{  240  -  255 blue & red scale with full green
            red := -1
            green := 63
            blue := 63
            SEQ i = 240 FOR 16
              SEQ
                red := red + 4
                set.colour (channel, i, red, green, blue)
                blue := blue - 4
            --}}}
        --}}}
        --}}}
        TRUE
          SKIP
:
--}}}
--{{{  set.timing
PROC set.timing( VAL INT16 width, height, frame.frequency,
                 VAL INT32 line.frequency, pixel.clock,
                 VAL BOOL  interlace )

  --{{{  variables
  INT AW, HBP, HFP, HS:
  INT horizontal.cells, flyback:
  INT AL, SL,  VFP, VS, VBP:
  --}}}
  SEQ
    --{{{  calculate horizontal timing
    horizontal.cells := INT((pixel.clock / line.frequency)>> 5)
    AW := ((INT width) >> 5)
    --{{{  COMMENT
    --:::A  0 0
    --{{{
    IF
      AW > (horizontal.cells - (horizontal.cells/5)) -- 80%
        STOP   -- Display set too wide
      TRUE
        SKIP
    --}}}
    --}}}
    flyback := horizontal.cells - AW
    HBP := flyback >> 1
    HFP := (flyback - HBP) >> 1
    HS :=  HBP - HFP
    --}}}
    --{{{  calculate vertical timing
    SL := INT (line.frequency  (INT32 frame.frequency))
```

```
      IF
        (INT height) > 1024
          AL := 1024
        TRUE
          AL := (INT height)
      --{{{  COMMENT test numbers
      --:::A  0 0
      --{{{  test numbers
      IF
        (INT AL) > (SL - (SL/5)) -- 80%
          STOP -- Image is set too tall for frame rate
        TRUE
          SKIP
      --}}}
      --}}}
      flyback := SL - (INT height)
      VFP := 3
      VS := 3
      VBP := flyback - 6
      --}}}
      --{{{  send video timing
      writeCRTC(CommandFIFO, CTRCReset)
      IF
        interlace
          writeCRTC(ParameterFIFO, #1B)
        TRUE
          writeCRTC(ParameterFIFO, #12)
      writeCRTC(ParameterFIFO, ((AW - 2) /\ #FE))
      writeCRTC(ParameterFIFO, (HS - 1) \/ ((VS /\ 7) << 5))
      writeCRTC(ParameterFIFO, ((VS /\ #18) >> 3) \/ ((HFP -1) << 2))
      writeCRTC(ParameterFIFO, (HBP - 1) /\ #3F)
      writeCRTC(ParameterFIFO, VFP /\ #3F)
      writeCRTC(ParameterFIFO, AL /\ #FF)
      writeCRTC(ParameterFIFO, ((AL /\ #0300) >> 8) \/ (VBP << 2))
      --}}}
      --{{{  unblank display & select mode
      writeCRTC(CommandFIFO, CTRCBctrl)
      ChannelModeSelect := 1
      --}}}
:
--}}}
--{{{  locals
INT16 width, height, frame.frequency:
INT32 line.frequency, pixel.clock:
BOOL interlace, running:
INT16 channel, pixel, red, green, blue, table:
--}}}
SEQ
  --{{{  command interpreter
  running := TRUE
  init.G170( INT channel.A, 0 )
  init.G170( INT channel.B, 0 )
  init.G170( INT channel.C, 0 )
  WHILE running
    command ? CASE
      crtc.init; width; height; line.frequency;
                 frame.frequency; pixel.clock; interlace
        set.timing( width, height, frame.frequency,
                    line.frequency, pixel.clock, interlace )
      crtc.color; channel; pixel; red; green; blue
        set.colour((INT channel), (INT pixel),
                   (INT red), (INT green), (INT blue))
      crtc.initLUT; channel; table
        init.G170((INT channel), (INT table))
```

```
          crtc.stop
            running := FALSE
      --}}}
:
--}}}
--}}}
```

6.2.1.3. PROC B409.stub "b409stub.occ"

```
PROC B409.stub ( CHAN OF ANY  in, out )

  #INCLUDE "crtc.inc"
  #USE "graphics.lib"
  B409( in )
:
```

6.2.1.4. PROC Background "backgrou.occ"

```
PROC Background ( CHAN OF ANY fromTarget, toSP,
                              fromPrev, toPrev, fromNext, toNext,
                  VAL INT position )

  #INCLUDE "s_header.inc"
  REAL32 g.scale :
  --{{{  constants
  VAL  packet.length   IS  16 :
  VAL  num.packets     IS  (128 * 8) / packet.length :

  VAL  min.signal      IS     0 :
  VAL  max.signal      IS 65535 :
  --}}}
  --{{{  ProcessRow
  PROC ProcessRow ( [packet.length] INT data,
                    [packet.length] REAL32    back.row,    gain.row,
offset.row )

    [] REAL32 target RETYPES data :
    SEQ
      SEQ i = 0 FOR packet.length
        INT digital :
        SEQ
          digital := INT TRUNC( (((target[i] + back.row[i]) *
                                  gain.row[i]) + offset.row[i]) * g.scale
)
          IF
            digital < min.signal
              data[i] := min.signal
            digital > max.signal
              data[i] := max.signal
            TRUE
              data[i] := digital
  :
  --}}}
  --{{{  ProcessFrame
  PROC ProcessFrame ( CHAN OF ANY in, out,
                      [128][8] REAL32 Background, Gain, Offset )

    --{{{  retype array to packet.length
      [num.packets][packet.length] REAL32      p.background      RETYPES
Background :
    [num.packets][packet.length] REAL32   p.gain            RETYPES Gain
:
    [num.packets][packet.length] REAL32   p.offset        RETYPES Offset
:
    --}}}
    --{{{  variables
    INT in.ptr, out.ptr, process.ptr, temp :
    [3][packet.length] INT buffer :
    PLACE buffer IN WORKSPACE :
    --}}}
    SEQ
      in.ptr := 2
      process.ptr := 1
      out.ptr := 0
      --{{{  get first row
      in ? buffer[0]
      --}}}
      --{{{  get second row and process first row
      PAR PAR
```

```
        in ? buffer[1]
        ProcessRow( buffer[0], p.background[1], p.gain[1], p.offset[1] )
      --}}}
      --{{{  do middle rows
      SEQ row = 1 FOR (num.packets-2)
        SEQ
          PRI PAR
            PAR
              in ? buffer[in.ptr]
              out ! buffer[out.ptr]
            ProcessRow( buffer[process.ptr], p.background[row],
                        p.gain[row], p.offset[row] )
          temp := out.ptr
          out.ptr := process.ptr
          process.ptr := in.ptr
          in.ptr := temp
      --}}}
      --{{{  process last row
      VAL i IS num.packets - 1 :
      PRI PAR
        out ! buffer[out.ptr]
        ProcessRow( buffer[process.ptr], p.background[i],
                    p.gain[i], p.offset[i] )
      --}}}
      --{{{  output last row
      out ! buffer[ process.ptr ]
      --}}}
  :
  --}}}
  --{{{  CalibrationFrame
  PROC CalibrationFrame ( CHAN OF ANY out, VAL REAL32 level,
                          [128][8] REAL32 Gain, Offset )

    --{{{  retype array to packet.length
    [num.packets][packet.length] REAL32   p.gain           RETYPES Gain
:
    [num.packets][packet.length] REAL32   p.offset         RETYPES Offset
:
    --}}}
    --{{{  variables
    INT out.ptr :
    [2][packet.length] INT buffer :
    PLACE buffer IN WORKSPACE :
    [packet.length] REAL32 b.row :
    [packet.length] INT t.row :
    --}}}
    SEQ
      out.ptr := 0
      --{{{  initialize
      VAL INT i.level RETYPES level :
      SEQ i = 0 FOR packet.length
        SEQ
          b.row[i] := 0.0 (REAL32)
          t.row[i] := i.level
      --}}}
      --{{{  process first row
      SEQ
        buffer[0] := t.row
        ProcessRow( buffer[0], b.row, p.gain[1], p.offset[1] )
      --}}}
      --{{{  do middle rows
      SEQ row = 1 FOR (num.packets-1)
        SEQ
          PRI PAR
```

```
            out ! buffer[out.ptr]
            SEQ
              buffer[1-out.ptr] := t.row
              ProcessRow( buffer[1-out.ptr], b.row,
                          p.gain[row], p.offset[row] )
          out.ptr := 1 - out.ptr
      --}}}
      --{{{  output last row
      out ! buffer[ out.ptr ]
      --}}}
:
--}}}
--{{{  SelectRow
PROC SelectRow ( [8] REAL32 dest, [128] REAL32 source )

  SEQ i = 0 FOR 8
    dest[i] := source[ (i*16) + position ]
:
--}}}
--{{{  variables
[max.frames][128][8] REAL32 Background :
[128][8] REAL32 Gain, Offset :

BYTE length :
[max.message] INT message :
command IS message[0] :
params IS [message FROM 1 FOR (max.message-1)] :
[] REAL32 r.params RETYPES params :
--}}}
SEQ
  --{{{  initialize last background frame
  SEQ i = 0 FOR 128
    SEQ j = 0 FOR 8
      Background[max.frames-1][i][j] := 0.0008234782 (REAL32)
  --}}}
  WHILE TRUE
    SEQ
      --{{{  get command and pass on
      fromPrev ? length::message
      IF
        position < 15
          toNext ! length::message
        TRUE
          SKIP
      --}}}
      CASE command
        --{{{  c.set.background
        c.set.background
          ProcessFrame( fromTarget, toSP, Background[ params[0] ],
                        Gain, Offset )
        --}}}
        --{{{  c.background.row
        c.background.row
          SelectRow( Background[ params[0] ][ params[1] ], [r.params
FROM 2 FOR 128] )
        --}}}
        --{{{  c.gain.row
        c.gain.row
          SelectRow( Gain[ params[0] ], [r.params FROM 1 FOR 128] )
        --}}}
        --{{{  c.offset.row
        c.offset.row
          SelectRow( Offset[ params[0] ], [r.params FROM 1 FOR 128] )
        --}}}
```

```
          --{{{  c.global.scale
          c.global.scale
            g.scale := r.params[0]
          --}}}
          --{{{  c.test.background
          c.test.background
            SEQ
              SEQ i = 0 FOR 128
                SEQ j = 0 FOR 8
                  VAL  col  IS  (j << 4) + position :
                  SEQ
                    Gain[i][j] := 0.80008787 (REAL32) -
                                  ((REAL32 ROUND   col)   /   254.3456
(REAL32))
                    Offset[i][j] := 0.0008723984 (REAL32) +
                                    ((REAL32 ROUND i) * 19.789 (REAL32))
                       Background[max.frames-1][i][j] :=    0.0008234782
(REAL32)
          --}}}
          --{{{  c.calibration.frame
          c.calibration.frame
            CalibrationFrame( toSP, r.params[0], Gain, Off. : )
          --}}}
:
```

PAGES 31 - 34 INTENTIONALLY OMITTED

TEXT IS COMPLETE

6.2.1.5. PROC Controller "controll.occ"

```
PROC Controller ( CHAN OF ANY fromHost, toHost, fromGTSEI, toGTSEI,
                              fromBG, toBG, fromSP, toSP )

  PRI PAR
    --{{{  make processing a high priority process
    #INCLUDE "s_header.inc"
    --{{{  variables
    --{{{  command variables
    BYTE length :
    [max.message] INT message :
    command IS message[0] :
    params IS [message FROM 1 FOR (max.message-1)] :
    [] REAL32 r.params RETYPES params :
    --}}}

    [max.sim.frames] REAL32 frame.rate, frame.time, frame.range :
    [max.sim.frames] INT  ticks :
    [max.sim.frames][p.length] REAL32 position :

    INT frames.loaded, start, value :   -- temporary variables
    INT current.frame, increment :
    INT offset, col, row :
    BOOL test.mode :
    INT calibration.frame, num.cal.frames :
    [10] REAL32 calibration.level :
    [10] INT    sp.cal.level :

    VAL  seconds.per.tick  IS  1.0E-6(REAL32) :
    TIMER clock :

    --{{{  force some scalars in vector space
    [3] INT frame.array :
    sim.frame   IS frame.array[0] :
    first.frame IS frame.array[1] :
    last.frame  IS frame.array[2]:
    --}}}
    --}}}
    SEQ
      --{{{  initialize
      current.frame := max.frames - 1
      increment := 1
      test.mode := TRUE
      calibration.frame := 10
      --}}}
      WHILE TRUE
        SEQ
          --{{{  get command
          fromHost ? length::message
          --}}}
          --{{{  process command
          IF
            --{{{  GTSEI and Target commands
            (command >= 256) AND (command < 768)
              toGTSEI ! length::message
            --}}}
            --{{{  Background commands
            (command >= 768) AND (command < 1024)
              toBG ! length::message
            --}}}
            --{{{  Guidance commands
            (command >= 1280) AND (command < 1536)
```

```
              toSP ! length::message
            --}}}
            --{{{  c.read.graphics
            command = c.read.graphics
              INT bufLength :
              INT number.of.transfers :
              [maxGraphicBuffer]BYTE graphicsBuffer :
              SEQ
                toSP ! length::message
                fromSP ? number.of.transfers
                toHost ! number.of.transfers
                SEQ i = 0 FOR number.of.transfers
                  SEQ
                    fromSP ? bufLength::graphicsBuffer
                    toHost ! bufLength::graphicsBuffer
            --}}}
            --{{{  c.frame.start
            command = c.start.frame
              SEQ
                IF
                  params[0] < 0
                    SKIP
                  params[0] = 0
                    sim.frame := params[1]
                  TRUE
                    sim.frame := first.frame + params[1]
                --{{{  sim.frame := MAX( 0, MIN( last.frame, sim.frame
))
                IF
                  sim.frame < 0
                    sim.frame := 0
                  sim.frame > last.frame
                    sim.frame := last.frame
                  TRUE
                    SKIP
                --}}}
                increment := params[2]
                --calibration.frame := num.cal.frames
                test.mode := FALSE
            --}}}
            --{{{  c.run.single
            command = c.run.single
              SEQ
                IF
                  calibration.frame < num.cal.frames
                    --{{{  send calibration frame
                    SEQ
                         toBG !     BYTE     2;     c.calibration.frame;
calibration.level[ calibration.frame ]
                      toSP ! BYTE 9; c.sp.frame; calibration.frame;
                                     sp.cal.level[ calibration.frame ];
0; 65535;
                                        0.0(REAL32); 0.0(REAL32);    -1;
64.0(REAL32)
                      calibration.frame := calibration.frame + 1
                    --}}}
                  sim.frame < first.frame
                    --{{{  send next non-FPA frame
                    SEQ
                      toSP ! 12(BYTE); c.guidance.run; 0; frame.range[
sim.frame ];
                             frame.time[ sim.frame ]; 0; 0; position[
sim.frame ]
```

```
                    toSP ! 6(BYTE); c.display.info; 0; frame.range[
sim.frame ];
                           frame.time[ sim.frame ]; 0; 0

                  IF
                    test.mode
                        frame.time[ sim.frame   ]   :=   frame.time[
sim.frame ] + (1.0(REAL32) /
                                                    frame.rate[
sim.frame ])
                    sim.frame < last.frame
                      sim.frame := sim.frame + increment
                    TRUE
                      SKIP
                --}}}
              TRUE
                --{{{  send next FPA frame
                SEQ
                  current.frame := sim.frame - first.frame

                  offset := ((row /\ 3) << 2) + (col /\ 3)
                  toGTSEI ! BYTE 2; c.set.crossbar; (col >> 2) /\ 15
                   toGTSEI !  BYTE  5;  c.set.target;  current.frame;
offset; (row>>2); (col>>2)
                  toBG ! BYTE 2; c.set.background; current.frame
                  toSP ! BYTE 15; c.sp.frame; -1; 0; 2500; 65535;
                                       frame.range[ sim.frame      ];
frame.time[ sim.frame ];
                                         current.frame+1; frame.rate[
sim.frame ]; position[ sim.frame]
                  IF
                    test.mode
                      --{{{  update statistics
                      SEQ
                          frame.time[ sim.frame   ]   :=   frame.time[
sim.frame ] + (1.0(REAL32) /
                                                      frame.rate[
sim.frame ])
                             frame.range[ sim.frame  ]  :=  frame.range[
sim.frame ] -
                                                      (10000.0(REAL32)
/ frame.rate[ sim.frame ])
                      --}}}
                    sim.frame < last.frame
                      sim.frame := sim.frame + increment
                    TRUE
                      SKIP
                --}}}
            --{{{  process any guidance commands
             VAL   delay.ticks    IS    INT  ROUND  (0.05  (REAL32)  /
seconds.per.tick) :
            INT time.now, interrupt.time :
            INT command :
            BOOL exit :
            SEQ
              clock ? time.now
              interrupt.time := time.now + delay.ticks

              exit := FALSE
              WHILE NOT exit
                PRI ALT
                  clock ? AFTER interrupt.time
                    exit := TRUE
                  fromSP ? command
```

```
                        --{{{  process command
                        CASE command
                          cc.shift.image
                            INT shift.col, shift.row :
                            SEQ
                              fromSP ? shift.col; shift.row
                              col := (col + shift.col) /\ 511
                              row := (row + shift.row) /\ 511
                          ELSE
                            SKIP
                        --}}}
                --}}}
              --}}}
              --{{{  c.run.continuous
              command = c.run.continuous
                 VAL   delay.ticks   IS   INT   ROUND(   5.0E-4(REAL32)   /
seconds.per.tick ) :
                INT last.start.time, next.start.time, interrupt.time :
                INT command :
                BOOL running, exit :
                SEQ
                  --{{{  check sending calibration frames
                  WHILE calibration.frame < num.cal.frames
                    --{{{  send calibration frame
                    SEQ
                          toBG !     BYTE     2;     c.calibration.frame;
calibration.level[ calibration.frame ]
                      toSP ! BYTE 9; c.sp.frame; calibration.frame;
                                     sp.cal.level[ calibration.frame ]; 0;
65535;
                                         0.0(REAL32); 0.0(REAL32);     -1;
64.0(REAL32)
                      calibration.frame := calibration.frame + 1
                    --}}}
                  --}}}
                  --{{{  initialize
                  clock ? last.start.time
                  interrupt.time := last.start.time
                  next.start.time := interrupt.time + delay.ticks
                  --}}}
                  running := TRUE
                  WHILE running
                    SEQ
                      --{{{  send current frame
                      IF
                        sim.frame < first.frame
                          --{{{  send next non-FPA frame
                          SEQ
                            --{{{  wait for correct time
                            INT current.time :
                            VAL  wait.ticks  IS  INT ROUND( 1.0E-4(REAL32)
/ seconds.per.tick ) :
                            SEQ
                              clock ? current.time
                              IF
                                  (next.start.time MINUS  current.time)   >
wait.ticks
                                    clock ? AFTER next.start.time
                                TRUE
                                  SKIP
                            --}}}

                              toSP !    12(BYTE);    c.guidance.run;    0;
frame.range[ sim.frame ];
```

```
                                    frame.time[ sim.frame    ];    0;    0;
position[ sim.frame ]
                              toSP !    6(BYTE);    c.display.info;    0;
frame.range[ sim.frame ];
                                    frame.time[ sim.frame ]; 0; 0
                        --}}}
                      TRUE
                        --{{{  send out FPA frame
                        SEQ
                          current.frame := sim.frame - first.frame

                          offset := ((row /\ 3) << 2) + (col /\ 3)

                          toGTSEI ! BYTE 2; c.set.crossbar; (col >> 2)
/\ 15
                          toGTSEI ! BYTE 5; c.set.target; current.frame;
offset; (row>>2); (col>>2)

                          --{{{  wait for correct time
                          INT current.time :
                          VAL  wait.ticks  IS  INT ROUND( 1.0E-4(REAL32)
/ seconds.per.tick ) :
                          SEQ
                            clock ? current.time
                            IF
                                (next.start.time MINUS  current.time)  >
wait.ticks
                                  clock ? AFTER next.start.time
                              TRUE
                                SKIP
                          --}}}

                          toBG ! BYTE 2; c.set.background; current.frame
                           toSP !  BYTE  15;  c.sp.frame;  -1;  0;  2500;
65535;
                                                frame.range[ sim.frame    ];
frame.time[ sim.frame ];
                                                current.frame+1; frame.rate[
sim.frame ];
                                             position[ sim.frame ]

                          --}}}
                    --{{{  move to next frame
                    IF
                      test.mode
                        --{{{  update statistics
                        SEQ
                          frame.time[ sim.frame    ]    :=    frame.time[
sim.frame ] + (1.0(REAL32) /
                                                          frame.rate[
sim.frame ])
                            frame.range[ sim.frame    ]    :=    frame.range[
sim.frame ] -
                                                           (10000.0(REAL32) /
frame.rate[ sim.frame ])
                        --}}}
                      sim.frame < last.frame
                        sim.frame := sim.frame + increment
                      TRUE
                        SKIP
                    --}}}
                    --{{{  update for next frame
                    last.start.time := next.start.time
                    --clock ? last.start.time
```

```
                    next.start.time :=  last.start.time  PLUS  ticks[
sim.frame ]
                  interrupt.time := next.start.time MINUS delay.ticks
                  --}}}
                  --}}}
                  exit := FALSE
                  WHILE NOT exit
                    PRI ALT
                      clock ? AFTER interrupt.time
                        exit := TRUE
                      fromHost ? command
                        --{{{  process command
                        CASE command
                          cc.shift.image
                            INT shift.col, shift.row :
                            SEQ
                              fromHost ? shift.col; shift.row
                              col := (col + shift.col) /\ 511
                              row := (row + shift.row) /\ 511
                          cc.exit
                            SEQ
                              exit := TRUE
                              running := FALSE
                          ELSE
                            SKIP
                        --}}}
                      fromSP ? command
                        --{{{  process command
                        CASE command
                          cc.shift.image
                            INT shift.col, shift.row :
                            SEQ
                              fromSP ? shift.col; shift.row
                              col := (col + shift.col) /\ 511
                              row := (row + shift.row) /\ 511
                          ELSE
                            SKIP
                        --}}}
            --}}}
            --{{{  c.frame.rate
            command = c.frame.rate
              SEQ
                start := params[0]
                frames.loaded := params[1]
                [frame.rate FROM start FOR frames.loaded] :=
                                                [r.params FROM   2   FOR
frames.loaded]
            --}}}
            --{{{  c.frame.time
            command = c.frame.time
              SEQ
                start := params[0]
                frames.loaded := params[1]
                [frame.time FROM start FOR frames.loaded] :=
                                                [r.params FROM  2  FOR
frames.loaded]
            --}}}
            --{{{  c.frame.range
            command = c.frame.range
              SEQ
                start := params[0]
                frames.loaded := params[1]
                [frame.range FROM start FOR frames.loaded] :=
```

```
                                                    [r.params FROM  2  FOR
frames.loaded]
            --}}}
            --{{{  c.sim.position
            command = c.sim.position
              INT ptr :
              SEQ
                value := params[0]
                start := params[1]
                frames.loaded := params[2]

                ptr := 3
                SEQ i = start FOR frames.loaded
                  SEQ
                    position[i][value] := r.params[ptr]
                    ptr := ptr + 1
            --}}}
            --{{{  c.sim.start.frames
            command = c.sim.start.frames
              SEQ
                first.frame := params[0]
                last.frame  := params[1]

                IF
                  (first.frame + 1) < last.frame
                    SEQ
                      SEQ i = 0 FOR last.frame
                         ticks[i] :=  INT  ROUND(  (frame.time[i+1]  -
frame.time[i]) /
                                                  seconds.per.tick )
                      ticks[ last.frame ] := ticks[ last.frame-1 ]
                  TRUE
                    SKIP
            --}}}
            --{{{  c.test.controller
            command = c.test.controller
              SEQ
                sim.frame := first.frame + (max.frames - 1)
                test.mode := TRUE
                row := 0
                col := 0
                 ticks[ sim.frame  ]  :=  INT  ROUND(  (  1.0(REAL32)  /
64.0(REAL32) ) /
                                                      seconds.per.tick )
                frame.range[ sim.frame ] := 100000.0 (REAL32)
                frame.time[ sim.frame ] := 0.0 (REAL32)
                frame.rate[ sim.frame ] := 64.0 (REAL32)
            --}}}
            --{{{  c.restart
            command = c.restart
              SEQ
                calibration.frame := 0
                test.mode := FALSE
                sim.frame := 0
                row := 0
                col := 0
            --}}}
            --{{{  c.set.calibration
            command = c.set.calibration
              SEQ
                num.cal.frames := params[0]
                [calibration.level FROM 0 FOR num.cal.frames] :=
                        [r.params FROM 1 FOR num.cal.frames]
                [sp.cal.level FROM 0 FOR num.cal.frames] :=
```

```
                                [params FROM      1+num.cal.frames      FOR
num.cal.frames]
            --}}}
            --{{{  else SKIP
            TRUE
              SKIP
            --}}}
          --}}}
    --}}}
    SKIP
:
```

6.2.1.6. PROC Firstbuffer (Graphics Buffer) "firstbuf.occ"

```
PROC FirstBuffer ( CHAN OF ANY in, out, fromNext, toPrev,
                   VAL INT position, shift )

  --{{{  variables
  [2][64] INT input.buffer :
  INT count :
  --}}}
  --{{{  channels
  CHAN OF ANY synch0, synch1, internal :
  --}}}
  --{{{  Receiver
  PROC Receiver ( CHAN OF ANY in, out0, out1, [2][64] INT buffer )

    INT i :
    SEQ
      i := 0
      WHILE TRUE
        SEQ
          in ? buffer[i]
          out0 ! i
          out1 ! i
          i := 1 - i
  :
  --}}}
  --{{{  Sender
  PROC Sender ( CHAN OF ANY in, out, [2][64] INT buffer )

    INT i :
    SEQ
      WHILE TRUE
        SEQ
          in ? i
          out ! buffer[i]
  :
  --}}}
  --{{{  Extractor
  PROC Extractor ( CHAN OF ANY internal, in, out,
                   VAL INT count )

    --{{{  variables
    [2][64][2] BYTE buffer :
    INT output :
    --}}}
    SEQ
      internal ? buffer[0]
      output := 0
      WHILE TRUE
        SEQ
          SEQ i = 0 FOR count
            SEQ
              PAR
                out ! buffer[output]
                in ? buffer[ 1-output ]
              output := 1 - output
          PAR
            out ! buffer[output]
            internal ? buffer[1-output]
          output := 1 - output
  :
  --}}}
  --{{{  Formatter
```

```
  PROC Formatter ( CHAN OF ANY synch, out,
                   [2][64] INT input.buffer )

    --{{{  variables
    [2][64][4] BYTE b.in RETYPES input.buffer :
    [64][2] BYTE buffer :
    [64*2] BYTE buffer1 RETYPES buffer :
    INT in.ptr :
    --}}}
    SEQ
      WHILE TRUE
        SEQ
          --{{{  form message in buffer
          SEQ
            synch ? in.ptr

            source IS input.buffer[in.ptr] :
            INT p :
            SEQ
              p := 0
              SEQ i = 0 FOR 64
                INT store :
                SEQ
                  store := source[i] >> shift
                  --{{{  check for zeroing store
                  IF
                    store = 0
                      IF
                        source[i] <> 0
                          store := 1
                        TRUE
                          SKIP
                    TRUE
                      SKIP
                  --}}}
                  buffer1[p] := BYTE store
                  buffer1[p+1] := BYTE store
                  p := p + 2
          --}}}
          out ! buffer
  :
  --}}}
  SEQ
    IF
      position < 8
        count := 7 - position
      TRUE
        count := 15 - position
    PRI PAR
      PAR
        Receiver( in, synch0, synch1, input.buffer )
        Sender( synch0, out, input.buffer )
        Extractor( internal, fromNext, toPrev, count )
      Formatter( synch1, internal, input.buffer )
:
```

6.2.1.7. PROC Formatter "formatte.occ"

```
PROC Formatter ( CHAN OF ANY in, out )

  --{{{  constants
  VAL buffer.size IS 64*16 :
  --}}}
  --{{{  variables
  [2][buffer.size] BYTE input.buffer, output.buffer :
  [8] INT store.offset :
  --}}}
  --{{{  channels
  CHAN OF ANY synch0, synch1 :
  --}}}
  --{{{  Receiver
  PROC Receiver ( CHAN OF ANY in, out, [2][buffer.size] BYTE buffer )

    INT i :
    SEQ
      i := 0
      WHILE TRUE
        SEQ
          in ? buffer[i]
          out ! i
          i := 1 - i
  :
  --}}}
  --{{{  Formatter
  PROC Formatter ( CHAN OF ANY in, out,
                   [2][buffer.size] BYTE in.buffer, out.buffer )

    INT out.ptr, in.ptr :
    SEQ
      out.ptr := 0
      WHILE TRUE
        SEQ
          in ? in.ptr
          --{{{  format
          [8][64][2] BYTE inb RETYPES in.buffer[in.ptr] :
          [64][16] BYTE outb RETYPES out.buffer[out.ptr] :
          SEQ
            SEQ i = 0 FOR 8
              source IS inb[i] :
              VAL start  IS i << 1 :
              MOVE2D( source, 0, 0, outb, start, 0, 2, 64 )
          --}}}
          out ! out.ptr
          out.ptr := 1 - out.ptr
  :
  --}}}
  --{{{  Sender
  PROC Sender ( CHAN OF ANY in, out, [2][buffer.size] BYTE buffer )

    INT i :
    SEQ
      WHILE TRUE
        SEQ
          in ? i
          out ! buffer[i]
  :
  --}}}
  SEQ
    WHILE TRUE
```

```
      PRI PAR
        PAR
          Receiver( in, synch0, input.buffer )
          Sender( synch1, out, output.buffer )
        Formatter( synch0, synch1, input.buffer, output.buffer )
:
```

6.2.1.8. Various graphics routines "g_line.occ"

```
--{{{  SC line
--:::A  3 10
--{{{  line
--{{{  libraries
#INCLUDE "g_header.inc"
--}}}
--{{{  plot
PROC plot ( VAL [] INT window, [] BYTE screen,
            VAL INT x, y, VAL BYTE color )

  -- plots a single point on the screen
  -- makes sure the pixels are actually in the window
  VAL pixels.line IS window[ w.pixels.line ] :
  VAL size.x      IS window[ w.size.x ] :
  VAL size.y      IS window[ w.size.y ] :
  SEQ
    IF
      (x < 0) OR (y < 0) OR (x >= size.x) OR (y >= size.y)
        SKIP
      TRUE
        screen[ (y * pixels.line) + x ] := color
:
--}}}
--{{{  draw.line
PROC draw.line ( VAL [] INT window, [] BYTE screen,
                 VAL INT x1, y1, x2, y2,
                 VAL BYTE color )

  --{{{  clip.line
  PROC clip.line (VAL INT x1, y1, x2, y2, INT result, VAL []INT window)
    -- decides whether a line is totally on or off screen/window
    --{{{  codes
    VAL code.centre IS #00 :        -- 0000
    VAL code.left   IS #01 :        -- 0001
    VAL code.right  IS #02 :        -- 0010
    VAL code.bottom IS #04 :        -- 0100
    VAL code.top    IS #08 :        -- 1000
    --}}}
    INT code1, code2 :
    --{{{  PROC check
    PROC check (VAL INT x, y, INT code)
      VAL x.max IS window[ w.size.x ] :
      VAL y.max IS window[ w.size.y ] :
      SEQ
        IF
          --{{{  x.min <= x < x.max
          (x >= 0) AND (x < x.max)
            code := code.centre
          --}}}
          --{{{  x < x.min
          x < 0
            code := code.left
          --}}}
          --{{{  x > x.max
          TRUE --x >= x.max
            code := code.right
          --}}}
        IF
          --{{{  y.min  = y < y.max
          (y >= 0) AND (y < y.max)
            SKIP
```

```
          --}}}
          --{{{  y < y.min
          y < 0
            code := code \/ code.top
          --}}}
          --{{{  y >= y.max
          TRUE --y > y.max
            code := code \/ code.bottom
          --}}}
    :
    --}}}
    SEQ
      check (x1, y1, code1)
      check (x2, y2, code2)
      IF
        --{{{  line lies entirely within window
        (code1 \/ code2) = 0
          result := in.range
        --}}}
        --{{{  line lies entirely outside window
        (code1 /\ code2) <> 0
          result := not.inrange
        --}}}
        --{{{  partially in window perhaps
        TRUE
          result := part.inrange
        --}}}
:
--}}}
--{{{  slow.draw.line
PROC slow.draw.line ( VAL [] INT window, [] BYTE screen,
                      VAL INT x1, y1, x2, y2,
                      VAL BYTE color )
  -- uses Bresenham's integer algorithm to calculate plotting points
  -- calls plot to draw actual pixels on the screen
  VAL pixels.line IS window[w.pixels.line] :
  INT dx, dy :
  INT two.dx, two.dy :
  INT error :
  SEQ
    dx := x2 - x1
    dy := y2 - y1
    IF
      (dx <> 0) OR (dy <> 0)
        SEQ
          --{{{  a line to draw
          IF
            --{{{  dy = 0                -- horizontal line
            dy = 0
              SEQ i = x1 FOR dx + 1
                plot (window, screen, i, y1, color)
            --}}}
            --{{{  dx = 0                -- vertical line
            dx = 0
              SEQ
                IF
                  dy > 0
                    SEQ i = y1 FOR dy + 1
                      plot (window, screen, x1, i, color)
                  TRUE
                    SEQ i = y2 FOR (-dy) + 1
                      plot (window, screen, x1, i, color)
            --}}}
            --{{{  dx <> 0  dy <> 0      -- diagonal line
```

```
TRUE
  INT x, y :
  INT delta.y :
  SEQ
    x := x1
    y := y1
    two.dx := dx + dx
    IF
      --{{{  dy > 0
      dy > 0
        SEQ
          two.dy := dy + dy
          IF
            --{{{  dy > dx
            dy > dx
              SEQ
                error := two.dx - dy
                --{{{  plot line
                SEQ i = 0 FOR dy + 1
                  SEQ
                    plot (window, screen, x, y, color )
                    IF
                      error >= 0
                        SEQ
                          x := x + 1
                          error := error - two.dy
                      TRUE
                        SKIP
                    y := y + 1
                    error := error + two.dx
                --}}}
            --}}}
            --{{{  dy <= dx
            TRUE
              SEQ
                error := two.dy - dx
                --{{{  plot line
                SEQ i = 0 FOR dx + 1
                  SEQ
                    plot (window, screen, x, y, color )
                    IF
                      error >= 0
                        SEQ
                          y := y + 1
                          error := error - two.dx
                      TRUE
                        SKIP
                    x := x + 1
                    error := error + two.dy
                --}}}
        --}}}
      --{{{  dy < 0
      TRUE
        SEQ
          dy := -dy
          two.dy := dy + dy
          IF
            --{{{  dy > dx
            dy > dx
              SEQ
                error := two.dx - dy
                --{{{  plot line
                SEQ i = 0 FOR dy + 1
```

```
                                    SEQ
                                      plot (window, screen, x, y, color)
                                      IF
                                        error >= 0
                                          SEQ
                                            x := x + 1
                                            error := error - two.dy
                                        TRUE
                                          SKIP
                                      y := y - 1
                                      error := error + two.dx
                                  --}}}
                              --}}}
                              --{{{  dy <= dx
                              TRUE
                                SEQ
                                  error := two.dy - dx
                                  SEQ i = 0 FOR dx + 1
                                    SEQ
                                      plot (window, screen, x, y, color)
                                      IF
                                        error >= 0
                                          SEQ
                                            y := y - 1
                                            error := error - two.dx
                                        TRUE
                                          SKIP
                                      x := x + 1
                                      error := error + two.dy
                              --}}}
                        --}}}
                  --}}}
              --}}}
          TRUE
            plot (window, screen, x1, y1, color)
  :
  --}}}
  --{{{  fast.draw.line
  PROC fast.draw.line (VAL [] INT window, [] BYTE screen,
                       VAL INT x1, y1, x2, y2,
                       VAL BYTE  color)

    -- uses Bresenham's integer algorithm to calculate plotting points
    -- points are in increasing values of x
    -- only called when the line is known to be on screen / in window
and
    -- the current pixel size is one
    INT dx, dy, two.dx, two.dy, delta.x, delta.y :
    INT error, pixel :
    VAL pixels.line IS window[ w.pixels.line ] :
    SEQ
      dx := x2 - x1                -- always zero or positive
      dy := y2 - y1
      pixel := (y1 * pixels.line) + x1
      IF
        (dx <> 0) OR (dy <> 0)
          SEQ
            --{{{  a line to draw
            IF
              --{{{  dy = 0                -- horizontal line
              dy = 0
                SEQ i = pixel FOR dx + 1
                  screen[i] := color
              --}}}
```

```
--{{{  dx = 0                    -- vertical line
dx = 0
  SEQ
    IF
      dy > 0
        SEQ i = 0 FOR dy + 1
          SEQ
            screen[pixel] := color
            pixel := pixel + pixels.line
      TRUE
        SEQ i = 0 FOR (-dy) + 1
          SEQ
            screen[pixel] := color
            pixel := pixel - pixels.line
--}}}
--{{{  dx <> 0 AND dy <> 0
TRUE
  INT delta.y :
  SEQ
    two.dx := dx + dx
    IF
      dy > 0
        delta.y := pixels.line
      TRUE
        SEQ
          dy := -dy
          delta.y := -pixels.line
    two.dy := dy + dy
    IF
      --{{{  dy > dx
      dy > dx
        SEQ
          error := two.dx - dy
          --{{{  plot line
          SEQ i = 0 FOR dy + 1
            SEQ
              screen[pixel] := color
              IF
                error >= 0
                  SEQ
                    pixel := pixel + 1
                    error := error - two.dy
                TRUE
                  SKIP
              pixel := pixel + delta.y
              error := error + two.dx
          --}}}
      --}}}
      --{{{  dy <= dx
      TRUE
        SEQ
          error := two.dy - dx
          --{{{  plot line
          SEQ i = 0 FOR dx + 1
            SEQ
              screen[pixel] := color
              IF
                error >= 0
                  SEQ
                    pixel := pixel + delta.y
                    error := error - two.dx
                TRUE
                  SKIP
              pixel := pixel + 1
```

```
                                error := error + two.dy
                            --}}}
                        --}}}
                    --}}}
                --}}}
          TRUE
            screen[pixel] := color
  :
  --}}}
  INT x3, y3, x4, y4 :
  INT result :
  SEQ
    --{{{  swap x1,y1 with x2,y2 if x1 > x2
    IF
      x1 > x2
        SEQ
          x3 := x2
          y3 := y2
          x4 := x1
          y4 := y1
      TRUE
        SEQ
          x3 := x1
          y3 := y1
          x4 := x2
          y4 := y2
    --}}}
    clip.line (x3, y3, x4, y4, result, window)
    IF
      (result = in.range)
        fast.draw.line (window, screen, x3, y3, x4, y4, color)
      (result = part.inrange) OR (result = in.range)
        slow.draw.line (window, screen, x3, y3, x4, y4, color)
      TRUE
        SKIP
:
--}}}
--{{{  draw.polyline
PROC draw.polyline ( VAL [] INT window, [] BYTE screen,
                     VAL [][2]INT points, VAL BYTE color)

  -- calls draw line to draw the lines
  INT x, y :
  SEQ
    x := points[0][0]
    y := points[0][1]
    SEQ i = 1 FOR (SIZE points) - 1
      VAL point IS points[i] :
      SEQ
        draw.line( window, screen, x, y, point[0], point[1], color )
        x := point[0]
        y := point[1]
:
--}}}
--{{{  draw.rectangle
PROC draw.rectangle ( VAL [] INT window, [] BYTE screen,
                      VAL [2][2]INT p, VAL BYTE color)

  -- calls draw line to draw the lines
  INT x, y :
  SEQ
    draw.line( window, screen, p[0][0], p[0][1], p[1][0], p[0][1], color
)
```

```
    draw.line( window, screen, p[1][0], p[0][1], p[1][0], p[1][1], color
)
    draw.line( window, screen, p[1][0], p[1][1], p[0][0], p[1][1], color
)
    draw.line( window, screen, p[0][0}, p[1][1], p[0][0], p[0][0], color
)
:
--}}}
--}}}
--}}}
```

6.2.1.9. Graphics system control routines "g_system.occ"

```
--{{{  SC system
--:::A  3 10
--{{{  system
--{{{  libraries
#INCLUDE "crtc.inc"
#INCLUDE "g_header.inc"
--}}}
--{{{  set.colour
PROC set.colour ( CHAN OF CRTC message,
                  VAL INT channel, pixel, red, green, blue )
  -- set up a colour in the G170 colour look up table
  SEQ
    message ! crtc.color; INT16 channel; INT16 pixel;
                          INT16 red; INT16 green; INT16 blue
:
--}}}
--{{{  set.timing
PROC set.timing( CHAN OF CRTC message,
                   VAL INT   width,height,   line.frequency,   frame.rate,
pixel.clock,
                 VAL BOOL interlace )

  SEQ
     message !   crtc.init;   INT16   width;   INT16   height;   INT32
line.frequency;
              INT16 frame.rate; INT32 pixel.clock; interlace
:
--}}}
--{{{  set.B408
PROC set.B408( VAL INT DS, IE, EM, OE, R )

  --{{{  system constants
  VAL bpw.shift IS 2 :
  VAL mint      IS MOSTNEG INT :

  VAL DisplayStart.address    IS (#00000000   >< mint) >> bpw.shift :
  VAL InterlaceEnable.address IS (#000C0000   >< mint) >> bpw.shift :
  VAL EventMode.address       IS (#00100000   >< mint) >> bpw.shift :
  VAL OutputEnable.address    IS (#00140000   >< mint) >> bpw.shift :
  VAL Ready.address           IS (#00040000   >< mint) >> bpw.shift :

  INT DisplayStart, InterlaceEnable, EventMode, OutputEnable, Ready :

  PLACE  DisplayStart      AT  DisplayStart.address    :
  PLACE  InterlaceEnable   AT  InterlaceEnable.address :
  PLACE  EventMode         AT  EventMode.address       :
  PLACE  OutputEnable      AT  OutputEnable.address    :
  PLACE  Ready             AT  Ready.address           :
  --}}}
  SEQ
    DisplayStart := DS
    InterlaceEnable := IE
    EventMode := EM
    OutputEnable := OE
    Ready := R
:
--}}}
--{{{  init.G170
PROC init.G170 (CHAN OF CRTC message, VAL INT channel, table)

  SEQ
```

```
    message ! crtc.initLUT; INT16 channel; INT16 table
:
--}}}
--{{{  clear.window
PROC clear.window (VAL [] INT window, [] BYTE screen)

  VAL size.x       IS  window[ w.size.x ] :
  VAL size.y       IS  window[ w.size.y ] :
  VAL pixels.line  IS  window[ w.pixels.line ] :
  VAL b.color      IS  BYTE window[ w.background.color ] :
  INT ptr :
  SEQ
    SEQ i = 0 FOR size.x
      screen[i] := b.color
    ptr := pixels.line
    SEQ i = 0 FOR size.y - 1
      SEQ
        [screen FROM ptr FOR size.x] := [screen FROM 0 FOR size.x]
        ptr := ptr + pixels.line
:
--}}}
--}}}
--}}}
```

6.2.1.10. Graphics text routines "g_text.occ"

```
--{{{  SC text
--:::A  3 10
--{{{  text
#INCLUDE "g_header.inc"
--{{{  FUNCTION GetINT
INT FUNCTION GetINT (VAL INT pointer, VAL [] INT table)
  INT return :
  VAL [] BYTE b.table RETYPES table :
  VALOF
    [4]BYTE return.b RETYPES return :
    SEQ
      return.b[0] := b.table[pointer]
      return.b[1] := b.table[pointer + 1]
      return.b[2] := b.table[pointer + 2]
      return.b[3] := b.table[pointer + 3]
    RESULT return
:
--}}}
--{{{  FUNCTION GetINT16
INT16 FUNCTION GetINT16 (VAL INT pointer, VAL [] INT table)
  INT16 return :
  VAL [] BYTE b.table RETYPES table :
  VALOF
    [2]BYTE return.b RETYPES return :
    SEQ
      return.b[0] := b.table[pointer]
      return.b[1] := b.table[pointer + 1]
    RESULT return
:
--}}}
--{{{  FUNCTION GetBYTE
BYTE FUNCTION GetBYTE (VAL INT pointer, VAL [] INT table)
  BYTE return :
  VAL [] BYTE b.table RETYPES table :
  VALOF
    SEQ
      return := b.table[pointer]
    RESULT return
:
--}}}
--{{{  get.font.spec
PROC get.font.spec ( VAL [] INT font, [fs.size] INT spec)

  SEQ
    spec[ fs.PixWidth   ] := INT (GetINT16 (dfPixWidth.p,  font))
    spec[ fs.PixHeight  ] := INT (GetINT16 (dfPixHeight.p, font))
    spec[ fs.FirstChar  ] := INT (GetBYTE  (dfFirstChar.p, font))
    spec[ fs.LastChar   ] := INT (GetBYTE  (dfLastChar.p,  font))
    spec[ fs.BitsOffset ] :=      GetINT   (dfBitsOffset.p,font)
:
--}}}
--{{{  scroll
PROC scroll ( VAL [] INT window, []BYTE screen,
              VAL INT jump.size )
  -- scrolls a screen or window by the required number of lines
(jump.size)
  VAL size.x      IS  window[ w.size.x ] :
  VAL size.y      IS  window[ w.size.y ] :
  VAL pixels.line IS  window[ w.pixels.line ] :
  VAL b.color     IS  BYTE window[ w.background.color ] :
  INT p1, p2 :
```

```
  IF
    (jump.size > 0) AND (jump.size < size.y)
      --{{{  scroll screen
      SEQ
        p1 := 0
        p2 := pixels.line * jump.size
        SEQ i = 0 FOR (size.y - jump.size)
          SEQ
            [screen FROM p1 FOR size.x] := [screen FROM p2 FOR size.x]
            p1 := p1 + pixels.line
            p2 := p2 + pixels.line
        SEQ i = 0 FOR size.x
          screen[ p1 + i ] := b.color
        p2 := p1 + pixels.line
        SEQ i = 0 FOR (jump.size - 1)
          SEQ
            [screen FROM p2 FOR size.x] := [screen FROM p1 FOR size.x]
            p2 := p2 + pixels.line
      --}}}
    jump.size > 0
      --{{{  clear screen
      SEQ

        SEQ i = 0 FOR size.x
          screen[ i ] := b.color
        p2 := pixels.line
        SEQ i = 0 FOR (jump.size - 1)
          SEQ
            [screen FROM p2 FOR size.x] := [screen FROM 0 FOR size.x]
            p2 := p2 + pixels.line
      --}}}
    TRUE
      SKIP
:
--}}}
--{{{  draw.char  v2.0
PROC draw.char ( [] INT window, [] BYTE screen,
                 VAL BYTE char,
                 VAL [] INT font, VAL [fs.size] INT spec )

  --{{{  constants
  VAL mask IS 1 << 7 :
  pixels.line IS window[ w.pixels.line ] :
  size.x      IS window[ w.size.x      ] :
  size.y      IS window[ w.size.y      ] :
  cursor.x    IS window[ w.cursor.x    ] :
  cursor.y    IS window[ w.cursor.y    ] :

  VAL [] BYTE b.font RETYPES font :
  --}}}
  --{{{  variables
  INT bit, pixel :
  INT bitmask :
  INT char.width, offset, PixWidthBytes :
  INT char.spacing :
  INT character :
  --}}}
  --{{{  line feed
  PROC line.feed ( )

    SEQ
      cursor.y := cursor.y + spec[ fs.PixHeight ]
      IF
        (cursor.y + spec[ fs.PixHeight ]) < size.y
```

```
          SKIP
        TRUE
          INT scroll.lines :
          SEQ
            scroll.lines := (spec[ fs.PixHeight ] - (size.y - cursor.y))
+ 1
            cursor.y := (size.y - spec[ fs.PixHeight ]) - 1
            scroll( window, screen, scroll.lines )
  :
  --}}}
  SEQ
    character := INT char
    IF
      char = '*n'
        line.feed ()
      char = '*c'
        --{{{  carriage return
        cursor.x := 0
        --}}}
       (character >= spec[ fs.FirstChar ]) AND (character <= spec[
fs.LastChar])
        SEQ
          character := character - spec[ fs.FirstChar ]
          --{{{  set font data
          IF
            spec[ fs.PixWidth ] <> 0                                  -- Fixed
Width
              SEQ
                PixWidthBytes  := (spec[ fs.PixWidth ] + 7) >> 3
                char.width     := spec[ fs.PixWidth ]
                offset           := (character * (PixWidthBytes * spec[
fs.PixHeight ])) +
                                    spec[ fs.BitsOffset ]
                char.spacing   := char.width
            TRUE                                                      --
Variable Width
              INT char.width.p, char.pointer.p :
              SEQ
                char.width.p   := CharTable.p + (character << 2)
                char.pointer.p := char.width.p + 2
                char.width     := INT(GetINT16(char.width.p, font))
                offset         := INT(GetINT16(char.pointer.p, font))
                PixWidthBytes  := (char.width + 7) >> 3
                char.spacing   := char.width + 1
          --}}}
          IF
            --{{{  char too big
            (char.width > size.x) OR (spec[ fs.PixHeight ] > size.y)
              SKIP
            --}}}
            --{{{  room to draw char
            BOOL delayed.crlf :
            TRUE
              SEQ
                delayed.crlf := FALSE
                IF
                  --{{{  room to draw whole char
                  ((cursor.x + char.spacing) < size.x) AND
                      ((cursor.y + spec[ fs.PixHeight ]) < size.y)
                    SKIP
                  --}}}
                  --{{{  room to draw but at end of line
                  ((cursor.x + char.spacing) = size.x) AND
                      ((cursor.y + spec[ fs.PixHeight ]) < size.y)
```

```
                    delayed.crlf := TRUE
                  --}}}
                  --{{{  we need carriage return - line feed
                  TRUE
                    SEQ
                      cursor.x := 0
                      line.feed ()
                  --}}}
                pixel := (cursor.y TIMES pixels.line) + cursor.x
                --{{{  plot foreground only
                VAL f.color IS BYTE window[ w.foreground.color ] :
                SEQ
                  SEQ i = 0 FOR spec[ fs.PixHeight ]
                    SEQ
                      --{{{  draw row
                      SEQ j = 0 FOR PixWidthBytes
                        SEQ
                          bitmask := mask
                              VAL this.byte      IS      INT      b.font[
offset+((spec[fs.PixHeight] TIMES j) + i) ] :
                          SEQ k = 0 FOR 8
                            SEQ
                              bit := this.byte /\ bitmask
                              IF
                                --{{{  leave background as it is
                                bit = 0
                                  SKIP
                                --}}}
                                --{{{  plot foreground bit
                                TRUE
                                  screen[ pixel ] := f.color
                                --}}}
                              pixel := pixel + 1
                              bitmask := bitmask >> 1
                      --}}}
                        pixel :=    (pixel    -    (PixWidthBytes<<3))    +
pixels.line

                  cursor.x := cursor.x + char.spacing
                --}}}
                IF
                  delayed.crlf
                    --{{{  we need carriage return - line feed
                    SEQ
                      cursor.x := 0
                      line.feed ()
                    --}}}
                  TRUE
                    SKIP
            --}}}
      TRUE
        SKIP
:
--}}}
--{{{  write.string v2.0
PROC write.string ( [] INT window, [] BYTE screen,
                    VAL [] BYTE string, VAL [] INT font )

  [fs.size] INT spec :
  SEQ
    get.font.spec( font, spec )
    SEQ i = 0 FOR (SIZE string)
      draw.char( window, screen, string[i], font, spec )
:
```

```
--}}}
--{{{  string.width
PROC string.width ( VAL [] INT font, VAL [] BYTE string, INT width )

  [fs.size] INT spec :
  SEQ
    get.font.spec( font, spec )
    width := 0
    SEQ i = 0 FOR SIZE string
      --{{{  add width for character[i]
      INT character :
      SEQ
        character := INT string[i]
        IF
          (character >= spec[ fs.FirstChar ]) AND (character <= spec[
fs.LastChar] )
            SEQ
              character := character - spec[ fs.FirstChar ]
              --{{{  determine width from font
              IF
                spec[ fs.PixWidth ] <> 0                                --
Fixed Width
                  width := width + spec[ fs.PixWidth ]
                TRUE                                                    --
Variable Width
                  INT char.width.p, char.pointer.p :
                  SEQ
                    char.width.p   := CharTable.p + (character << 2)
                    width := (width + 1) + (INT(GetINT16(char.width.p,
font)))
              --}}}
          TRUE
            SKIP
      --}}}
:
--}}}
--}}}
--}}}
```

6.2.1.11. GIF routines (save captured display images) "gif.occ"

```
#INCLUDE "hostio.inc"
#USE "hostio.lib"
PROTOCOL message IS INT; INT :

PROC Encode (CHAN OF SP fs, ts, VAL INT32 GIFFile,
             VAL INT MaxColumn, MaxRow, BitsPerPixel,
             VAL []INT Palette, VAL [][]INT Pixels)
  --{{{  GIF Encoder
  PROC Encoder (CHAN OF message out, VAL [][]INT pixels,
                VAL INT bitsPerPixel)

    VAL max.table.size IS (1 << 13) :
    VAL pixel.rows IS (SIZE pixels) :
    --{{{  PROC Clear
    PROC Clear (VAL INT bits.per.pixel, INT CodeSize, NextValidCode,
                MaxCode, [max.table.size]INT Child, Sibling)

      SEQ
        CodeSize := bits.per.pixel + 1
        NextValidCode :=(1 << bits.per.pixel) + 2
        MaxCode := 1 << (bits.per.pixel + 1)
        SEQ I = 0 FOR max.table.size
          SEQ
            Child [I] := 0
            Sibling [I] := 0
    :
    --}}}
    --{{{  variable declarations
    [max.table.size] INT child, sibling, shade :
    INT codeSize, clearCode, endCode,
        minCode, maxCode, nextValidCode,
        color, son, parent, maxColor, pixCol, pixRow, pixelColumnsM1 :
    --}}}
    SEQ
      maxColor := (1 << bitsPerPixel) - 1
      color := 0
      --{{{  Initialize
      SEQ i = 0 FOR max.table.size
        SEQ
          child [i] := 0
          sibling [i] := 0
      codeSize := bitsPerPixel + 1
      clearCode := 1 << bitsPerPixel
      endCode := clearCode + 1
      nextValidCode := endCode + 1
      maxCode := clearCode << 1
      --}}}
      out ! clearCode; codeSize
      IF
        (0 < pixel.rows)
          --{{{
          SEQ
            pixelColumnsM1 := SIZE pixels [0]
            IF
              (0 < pixelColumnsM1)
                SEQ
                  color := pixels [0][0]
                  pixelColumnsM1 := pixelColumnsM1 - 1
                  IF
                    (1 < pixelColumnsM1)
                      SEQ
```

```
                    pixRow := 0
                    pixCol := 1
                TRUE
                  SEQ
                    pixRow := 1
                    pixCol := 0
          TRUE
            color := maxColor + 2
      --}}}
    TRUE
      color := maxColor + 2
  parent := color
  WHILE (color <= maxColor)
    --{{{  Compress
    SEQ
      IF
        (pixRow < pixel.rows)
          --{{{
          SEQ
            color := pixels [pixRow][pixCol]
            IF
              (pixCol < pixelColumnsM1)
                pixCol := pixCol + 1
              TRUE
                SEQ
                  pixRow := pixRow + 1
                  pixCol := 0
          --}}}
        TRUE
          color := maxColor + 2
      son := child [parent]
      IF
        son <= 0
          --{{{  Parent has no son
          SEQ
            child [parent] := nextValidCode
            shade [nextValidCode] := color
            out ! parent; codeSize
            parent := color
            nextValidCode := nextValidCode + 1
          --}}}
        TRUE
          --{{{  Otherwise
          SEQ
            IF
              shade [son] = color
                parent := son  -- make new parent
              TRUE
                --{{{  son not right color
                BOOL looping :
                INT brother :
                SEQ
                  brother := son
                  looping := TRUE
                  WHILE looping
                    SEQ
                      IF
                        sibling [brother] > 0
                          --{{{  Brother has brother
                          SEQ
                            brother := sibling [brother]
                            IF
                              shade [brother] = color
                                SEQ
```

```
                                        looping := FALSE
                                        parent := brother
                                    TRUE
                                      SKIP
                                --}}}
                              TRUE
                                --{{{  No brother, so create one
                                SEQ
                                  looping := FALSE
                                  sibling [brother] := nextValidCode
                                  shade [nextValidCode] := color
                                  out ! parent; codeSize
                                  parent := color
                                  nextValidCode := nextValidCode + 1
                                --}}}
                        --}}}
                    --}}}
                --{{{  Change code size if required
                IF
                  nextValidCode > maxCode
                    IF
                      codeSize < 12
                        SEQ
                          codeSize := codeSize + 1
                          maxCode := maxCode << 1
                      TRUE
                        SEQ
                          out ! clearCode; codeSize
                          Clear (bitsPerPixel, codeSize, nextValidCode,
                                 maxCode, child, sibling)
                  TRUE
                    SKIP
                --}}}
              --}}}
            out ! endCode; codeSize
  :
  --}}}
  --{{{  so.fwrite.INT16
  --Writes 2-byte integer, LSB first
  PROC so.fwrite.INT16 (CHAN OF SP fs, ts, VAL INT32 StreamID,
                        VAL INT16 Value, BYTE Result)

    VAL msb IS #FF00 (INT16) :
    VAL lsb IS #00FF (INT16) :

    BYTE Result2 :
    VAL [2]BYTE array RETYPES Value :

    SEQ
      --so.fwrite.char (fs, ts, StreamID, BYTE (Value /\ lsb), Result)
      --so.fwrite.char (fs, ts, StreamID, BYTE ((Value /\ msb) >> 8),
Result2)
      --Result := BYTE ((INT Result) \/ (INT Result2))
      so.fwrite.string (fs, ts, StreamID, array, Result)
  :
  --}}}
  --{{{  WriteBlock
  PROC WriteBlock (CHAN OF SP fs, ts, VAL INT32 StreamID,
                   [255] BYTE Block, INT Length, BYTE Result)
    [256]BYTE buffer :
    INT length.written :
    SEQ
      buffer[0] := (BYTE Length)
      buffer FROM 1 FOR Length := [Block FROM 0 FOR Length]
```

```
      so.fwrite.string (fs, ts, StreamID,
                        [buffer FROM 0 FOR (Length+1)], Result)
      Length := 0
  :
  --}}}
  --{{{  BlockByte
  PROC BlockByte (CHAN OF SP fs, ts, VAL INT32 StreamID,
                  [255] BYTE  Block,  INT  Index,  VAL  BYTE  Data,  BYTE
Result)

    SEQ
      Block [Index] := Data
      Index := Index + 1
      IF
        Index = 255
          WriteBlock (fs, ts, StreamID, Block, Index, Result)
        TRUE
          SKIP
  :
  --}}}
  --{{{  Output process from Encoder
  PROC Output (CHAN OF message FromEncoder, VAL INT32 GIFFile,
               VAL INT MaxColumn, MaxRow, BitsPerPixel,
               VAL []INT Palette                                )
    --{{{  Constants
    VAL byte.mask   IS #FF (INT32) :
    VAL size.of.int IS 4 :
    VAL colors      IS (1  << BitsPerPixel) :
    VAL max.byte    IS 40 :
    VAL depth       IS 2 :
    VAL max.gray    IS (1 << BitsPerPixel) :
    VAL ppw         IS ((size.of.int * 8) / BitsPerPixel) :
    VAL wpsl        IS (MaxColumn / ppw) :
    VAL red         IS 0 :
    VAL green       IS 1 :
    VAL blue        IS 2 :

    VAL gif.signature IS "GIF87a" :
    VAL global.color.map  IS #80 :
    VAL color.res         IS ((depth - 1) << 4) :
    VAL bits              IS (BitsPerPixel + 1) :
    VAL screen.height     IS (INT16 MaxRow) :
    VAL screen.left       IS 0 (INT16) :
    VAL screen.top        IS 0 (INT16) :
    VAL screen.width      IS (INT16 MaxColumn) :
    VAL screen.descriptor IS
              (BYTE ((global.color.map \/ color.res) \/
                                    (BitsPerPixel - 1))) :
    VAL background IS 0 (BYTE) :
    VAL endCode    IS ((1 << BitsPerPixel) + 1) :
    --}}}

    BYTE Result :
    INT CodeSize, OutByte, Shift, Value :
    INT32 Out :
    [3] BYTE ColorValue :
    [255] BYTE OutBlock :

    SEQ
      OutByte := 0
      --{{{  GIF Signature
      so.fwrite.string (fs, ts, GIFFile, gif.signature, Result)
      --}}}
      --{{{  Screen Descriptor
```

```
so.fwrite.INT16 (fs, ts, GIFFile, screen.width, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.height, Result)
so.fwrite.char (fs, ts, GIFFile, screen.descriptor, Result)
so.fwrite.char (fs, ts, GIFFile, background, Result)
so.fwrite.char (fs, ts, GIFFile, 0 (BYTE), Result)
--}}}
--{{{  Global Color Map
VAL PaletteColors IS (SIZE Palette) :
SEQ I = 0 FOR colors
  SEQ
    IF
      (I < PaletteColors)
        SEQ
          ColorValue [blue] :=
            (BYTE ((INT32 Palette [I]) /\ byte.mask))
          ColorValue [green] :=
            (BYTE (((INT32 Palette [I]) >> 8) /\ byte.mask))
          ColorValue [red] :=
            (BYTE (((INT32 Palette [I]) >> 16) /\ byte.mask))
      TRUE
        SEQ
          ColorValue [blue] := 0 (BYTE)
          ColorValue [green] := 0 (BYTE)
          ColorValue [red] := 0 (BYTE)
    SEQ J = 0 FOR 3
      so.fwrite.char (fs, ts, GIFFile, ColorValue [J], Result)

--}}}
--{{{  Image Descriptor
so.fwrite.char (fs, ts, GIFFile, ',', Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.left, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.top, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.width, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.height, Result)
so.fwrite.char (fs, ts, GIFFile, 0 (BYTE), Result)
--}}}
--{{{  Raster Data
--First byte is bits per image pixel
so.fwrite.char (fs, ts, GIFFile, BYTE BitsPerPixel, Result)
--{{{  Get first code
FromEncoder ? Value; CodeSize
Shift := CodeSize
Out := (INT32 Value)
--}}}
--{{{  Accept and package codes until end of image
WHILE (Value <> endCode)
  SEQ
    --{{{  Write any finished bytes
    WHILE (Shift > 8)
      SEQ
        BlockByte (fs, ts, GIFFile, OutBlock, OutByte,
                                  BYTE (Out /\ byte.mask), Result)
        Out := Out >> 8
        Shift := Shift - 8
    --}}}
    --{{{  Add next code
    FromEncoder ? Value ; CodeSize
    Out := Out \/ ((INT32 Value) << Shift)
    Shift := Shift + CodeSize
    --}}}
--}}}
--{{{  Output remaining codes
WHILE (Shift > 0)
  SEQ
```

```
          BlockByte (fs, ts, GIFFile, OutBlock, OutByte,
                                  BYTE (Out /\ byte.mask), Result)
          Out := Out >> 8
          Shift := Shift - 8
        IF
          OutByte <> 0
            WriteBlock (fs, ts, GIFFile, OutBlock, OutByte, Result)
          TRUE
            SKIP

        --Raster data terminates with 0-byte block
        WriteBlock (fs, ts, GIFFile, OutBlock, OutByte, Result)
        --}}}
        --}}}
        --{{{  GIF Terminator
        so.fwrite.char (fs, ts, GIFFile, ';', Result)
        --}}}
    :
    --}}}

    CHAN OF message EncoderToOutput :
    PAR
      Encoder (Pixels, EncoderToOutput, BitsPerPixel)
      Output (EncoderToOutput, GIFFile,
              MaxColumn, MaxRow, BitsPerPixel, Palette)
:
```

6.2.1.12. Alternative GIF routines "gif02.occ"

```
#INCLUDE "hostio.inc"
#USE "hostio.lib"

PROC Encode (CHAN OF SP fs, ts, VAL INT32 GIFFile,
             VAL INT MaxColumn, MaxRow, BitsPerPixel,
             VAL []INT Palette, VAL [][]INT Pixels)
  --{{{  GIF Encoder
  PROC Encoder (CHAN OF INT in, out, CHAN OF INT size, VAL INT
bitsPerPixel)

    --{{{  putCode
    PROC putCode (CHAN OF INT out, VAL INT value)
      out ! value
    :
    --}}}

    VAL max.table.size IS (1 << 13) :

    --{{{  Clear
    PROC Clear (VAL INT bits.per.pixel, INT CodeSize, NextValidCode,
                MaxCode, [max.table.size] INT Child, Sibling)

      SEQ
        CodeSize := bits.per.pixel + 1
        NextValidCode :=(1 << bits.per.pixel) + 2
        MaxCode := 1 << (bits.per.pixel + 1)
        SEQ I = 0 FOR max.table.size
          SEQ
            Child [I] := 0
            Sibling [I] := 0
    :
    --}}}

    [max.table.size] INT child, sibling, shade :
    INT codeSize, clearCode, endCode,
        minCode, maxCode, nextValidCode,
        color, son, parent, maxColor :

    SEQ
      maxColor := (1 << bitsPerPixel) - 1
      color := 0

      --{{{  Initialize
      SEQ
        SEQ i = 0 FOR max.table.size
          SEQ
            child [i] := 0
            sibling [i] := 0
        codeSize := bitsPerPixel + 1
        clearCode := 1 << bitsPerPixel
        endCode := clearCode + 1
        nextValidCode := endCode + 1
        maxCode := clearCode << 1
      --}}}

      SEQ
        putCode (out, clearCode)
        size ! codeSize
        in ? color
        parent := color
        WHILE (color <= maxColor)
```

```
--{{{  Compress
SEQ
  in ? color
  son := child [parent]
  IF
    son <= 0
      --{{{  Parent has no son
      SEQ
        child [parent] := nextValidCode
        shade [nextValidCode] := color
        putCode (out, parent)
        size ! codeSize
        parent := color
        nextValidCode := nextValidCode + 1
      --}}}
    TRUE
      --{{{  Otherwise
      SEQ
        IF
          shade [son] = color
            parent := son  -- make new parent
          TRUE
            --{{{  son not right color
            BOOL looping :
            INT brother :
            SEQ
              brother := son
              looping := TRUE
              WHILE looping
                SEQ
                  IF
                    sibling [brother] > 0
                      --{{{  Brother has brother
                      SEQ
                        brother := sibling [brother]
                        IF
                          shade [brother] = color
                            SEQ
                              looping := FALSE
                              parent := brother
                          TRUE
                            SKIP
                      --}}}
                    TRUE
                      --{{{  No brother, so create one
                      SEQ
                        looping := FALSE
                        sibling [brother] := nextValidCode
                        shade [nextValidCode] := color
                        putCode (out, parent)
                        size ! codeSize
                        parent := color
                        nextValidCode := nextValidCode + 1
                      --}}}
            --}}}
      --}}}
  --{{{  Change code size if required
  IF
    nextValidCode > maxCode
      IF
        codeSize < 12
          SEQ
            codeSize := codeSize + 1
            maxCode := maxCode << 1
```

```
                    TRUE
                      SEQ
                        putCode (out, clearCode)
                        size ! codeSize
                        Clear (bitsPerPixel, codeSize, nextValidCode,
                               maxCode, child, sibling)
              TRUE
                SKIP
            --}}}
          --}}}
      putCode (out, endCode)
      size ! codeSize
  :
  --}}}

  --{{{  so.fwrite.INT16
  --Writes 2-byte integer. LSB first
  PROC so.fwrite.INT16 (CHAN OF SP fs, ts, VAL INT32 StreamID,
                        VAL INT16 Value, BYTE Result)

    VAL msb IS #FF00 (INT16) :
    VAL lsb IS #00FF (INT16) :

    BYTE Result2 :

    SEQ
      so.fwrite.char (fs, ts, StreamID, BYTE (Value /\ lsb), Result)
      so.fwrite.char (fs, ts, StreamID, BYTE ((Value /\ msb) >> 8),
Result2)
      Result := BYTE ((INT Result) \/ (INT Result2))
  :
  --}}}

  --{{{  WriteBlock
  PROC WriteBlock (CHAN OF SP fs, ts, VAL INT32 StreamID,
                   [255] BYTE Block, INT Length, BYTE Result)
    BYTE Result2 :

    SEQ
      so.fwrite.char (fs, ts, StreamID, BYTE Length, Result)
      SEQ I = 0 FOR Length
        SEQ
          so.fwrite.char (fs, ts, StreamID, Block [I], Result2)
          Result := BYTE ((INT Result) \/ (INT Result2))
      Length := 0
  :
  --}}}

  --{{{  BlockByte
  PROC BlockByte (CHAN OF SP fs, ts, VAL INT32 StreamID,
                  [255] BYTE Block, INT Index, VAL BYTE Data, BYTE
Result)

    SEQ
      Block [Index] := Data
      Index := Index + 1
      IF
        Index = 255
          WriteBlock (fs, ts, StreamID, Block, Index, Result)
        TRUE
          SKIP
  :
  --}}}
```

```
--{{{  Input
--Input process for Encoder
PROC Input (CHAN OF INT ToEncoder, VAL [][]INT Pixels, VAL INT End)
  SEQ
    SEQ I = 0 FOR (SIZE Pixels)
      SEQ J = 0 FOR (SIZE Pixels [I])
        ToEncoder ! Pixels [I][J]
    ToEncoder ! End
:
--}}}

--{{{  Output
--Output process from Encoder
PROC Output (CHAN OF INT FromEncoder, Size, VAL INT32 GIFFile,
             VAL INT MaxColumn, MaxRow, BitsPerPixel,
             VAL []INT Palette                                  )
  --{{{  Constants
  VAL byte.mask   IS #FF (INT32) :
  VAL size.of.int IS 4 :
  VAL colors      IS (1  << BitsPerPixel) :
  VAL max.byte    IS 40 :
  VAL depth       IS 2 :
  VAL max.gray    IS (1 << BitsPerPixel) :
  VAL ppw         IS ((size.of.int * 8) / BitsPerPixel) :
  VAL wpsl        IS (MaxColumn / ppw) :
  VAL red         IS 0 :
  VAL green       IS 1 :
  VAL blue        IS 2 :

  VAL gif.signature IS "GIF87a" :
  VAL global.color.map  IS #80 :
  VAL color.res         IS ((depth - 1) << 4) :
  VAL bits              IS (BitsPerPixel + 1) :
  VAL screen.height     IS (INT16 MaxRow) :
  VAL screen.left       IS 0 (INT16) :
  VAL screen.top        IS 0 (INT16) :
  VAL screen.width      IS (INT16 MaxColumn) :
  VAL screen.descriptor IS
           (BYTE ((global.color.map \/ color.res) \/
                                  (BitsPerPixel - 1))) :
  VAL background IS 0 (BYTE) :
  VAL endCode    IS ((1 << BitsPerPixel) + 1) :
  --}}}

  BYTE Result :
  INT CodeSize, OutByte, Shift, Value :
  INT32 Out :
  [3] BYTE ColorValue :
  [255] BYTE OutBlock :

  SEQ
    OutByte := 0
    --{{{  GIF Signature
    so.fwrite.string (fs, ts, GIFFile, gif.signature, Result)
    --}}}

    --{{{  Screen Descriptor
    so.fwrite.INT16 (fs, ts, GIFFile, screen.width, Result)
    so.fwrite.INT16 (fs, ts, GIFFile, screen.height, Result)
    so.fwrite.char (fs, ts, GIFFile, screen.descriptor, Result)
    so.fwrite.char (fs, ts, GIFFile, background, Result)
    so.fwrite.char (fs, ts, GIFFile, 0 (BYTE), Result)
    --}}}
```

```
--{{{  Global Color Map
VAL PaletteColors IS (SIZE Palette) :
SEQ I = 0 FOR colors
  SEQ
    IF
      (I < PaletteColors)
        SEQ
          ColorValue [blue] :=
            (BYTE ((INT32 Palette [I]) /\ byte.mask))
          ColorValue [green] :=
            (BYTE (((INT32 Palette [I]) >> 8) /\ byte.mask))
          ColorValue [red] :=
            (BYTE (((INT32 Palette [I]) >> 16) /\ byte.mask))
      TRUE
        SEQ
          ColorValue [blue] := 0 (BYTE)
          ColorValue [green] := 0 (BYTE)
          ColorValue [red] := 0 (BYTE)
    SEQ J = 0 FOR 3
      so.fwrite.char (fs, ts, GIFFile, ColorValue [J], Result)

--}}}

--{{{  Image Descriptor
so.fwrite.char (fs, ts, GIFFile, ',', Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.left, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.top, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.width, Result)
so.fwrite.INT16 (fs, ts, GIFFile, screen.height, Result)
so.fwrite.char (fs, ts, GIFFile, 0 (BYTE), Result)
--}}}

--{{{  Raster Data
--First byte is bits per image pixel
so.fwrite.char (fs, ts, GIFFile, BYTE BitsPerPixel, Result)

--{{{  Get first code
FromEncoder ? Value
Size ? CodeSize
Shift := CodeSize
Out := (INT32 Value)
--}}}

--{{{  Accept and package codes until end of image
WHILE (Value <> endCode)
  SEQ
    --{{{  Write any finished bytes
    WHILE (Shift > 8)
      SEQ
        BlockByte (fs, ts, GIFFile, OutBlock, OutByte,
                                  BYTE (Out /\ byte.mask), Result)
        Out := Out >> 8
        Shift := Shift - 8
    --}}}

    --{{{  Add next code
    FromEncoder ? Value
    Size ? CodeSize
    Out := Out \/ ((INT32 Value) << Shift)
    Shift := Shift + CodeSize
    --}}}
--}}}

--{{{  Output remaining codes
```

```
      WHILE (Shift > 0)
        SEQ
          BlockByte (fs, ts, GIFFile, OutBlock, OutByte,
                                  BYTE (Out /\ byte.mask), Result)
          Out := Out >> 8
          Shift := Shift - 8
      IF
        OutByte <> 0
          WriteBlock (fs, ts, GIFFile, OutBlock, OutByte, Result)
        TRUE
          SKIP

      --Raster data terminates with 0-byte block
      WriteBlock (fs, ts, GIFFile, OutBlock, OutByte, Result)
      --}}}
      --}}}

      --{{{  GIF Terminator
      so.fwrite.char (fs, ts, GIFFile, ';', Result)
      --}}}
  :
  --}}}

  VAL end IS ((1 << BitsPerPixel) + 1) :

  CHAN OF INT ToEncoder, FromEncoder :
  CHAN OF INT Size :


  PAR
    Input (ToEncoder, Pixels, end)
    Encoder (ToEncoder, FromEncoder, Size, BitsPerPixel)
    Output (FromEncoder, Size, GIFFile,
            MaxColumn, MaxRow, BitsPerPixel, Palette)
:
```

6.2.1.13. PROC GTSEI "gtsei.occ"

```
PROC GTSEI ( CHAN OF ANY   fromController, toController, fromTarget,
toTarget,
                          toCrossbar0, toCrossbar1 )

  #INCLUDE "s_header.inc"
  --{{{  Table
  VAL Table IS [ [  0,  4 ],
                 [  2,  5 ],
                 [  1,  6 ],
                 [  7,  3 ],
                 [ 29, 31 ],
                 [ 30, 28 ],
                 [ 24, 25 ],
                 [ 27, 26 ],
                 [ 17, 19 ],
                 [ 23, 22 ],
                 [ 21, 16 ],
                 [ 20, 18 ],
                 [ 10,  8 ],
                 [ 13,  9 ],
                 [ 14, 12 ],
                 [ 11, 15 ] ] :
  --}}}
  --{{{  variables
  BYTE synch :
  BOOL target.wait :
  [16][3] BYTE Set0, Set1 :
  INT Current.Selection :
  INT Selection :
  PLACE Selection AT  #800 :

  BYTE length :
  [max.message] INT32 message :
  command IS message[0] :
  params IS [message FROM 1 FOR (max.message-1)] :
  --}}}
  --{{{  DetermineSetting
  PROC DetermineSetting ( [16][3] BYTE Set0, Set1, VAL INT offset )

    SEQ
      SEQ i = 0 FOR 16
        VAL connection IS BYTE Table[ ((i + offset) /\ 15) ][ 0 ]  :
        SEQ
          Set0[i][1] := connection
          Set1[i][2] := connection
  :
  --}}}
  SEQ
    --{{{  initialize
    SEQ i = 0 FOR 16
      SEQ
        Set0[i][0] := 0 (BYTE)
        Set0[i][1] := BYTE Table[i][0]
        Set0[i][2] := BYTE Table[i][1]
        Set1[i][0] := 0 (BYTE)
        Set1[i][1] := BYTE Table[i][1]
        Set1[i][2] := BYTE Table[i][0]

    toCrossbar0 ! 4(BYTE); Set0; 3(BYTE)
    toCrossbar1 ! 4(BYTE); Set1; 3(BYTE)
    --}}}
```

```
    target.wait := FALSE
    WHILE TRUE
      SEQ
        fromController ? length::message
        CASE INT command
          --{{{  c.set.crossbar
          c.set.crossbar
            DetermineSetting( Set0, Set1, INT params[0] )
          --}}}
          --{{{  c.set.target
          c.set.target
            SEQ
              IF
                target.wait
                  fromTarget ? synch
                TRUE
                  SKIP
              PAR
                toCrossbar0 ! 4(BYTE); Set0; 3(BYTE)
                toCrossbar1 ! 4(BYTE); Set1; 3(BYTE)
              toTarget ! length::message
              target.wait := TRUE
          --}}}
          --{{{  c.target.row
          c.target.row
            SEQ
              IF
                target.wait
                  fromTarget ? synch
                TRUE
                  SKIP
              target.wait := FALSE
              toTarget ! length::message
          --}}}
:
```

6.2.1.14. PROC Guidance "guidance.occ"

```
PROC Guidance ( CHAN OF ANY fromSP, toSP,
                            fromXBar, toXBar )

  --{{{  libraries
  #INCLUDE "s_header.inc"
  --}}}
  --{{{  constants
  VAL min.time IS 0.99 (REAL32) :
  --}}}
  --{{{  PROC test.setup
  PROC test.setup (CHAN OF ANY from.master, to.master)
    --{{{  constants
    VAL min.shift.x IS -8 :
    VAL max.shift.x IS  8 :
    VAL min.shift.y IS -8 :
    VAL max.shift.y IS  8 :

    VAL []REAL32 begin.divert IS [20000.0 (REAL32), 90000.0 (REAL32)] :
    VAL []REAL32 end.divert   IS [30000.0 (REAL32), 100000.0 (REAL32)] :

    VAL x.center IS 256 :
    VAL y.center IS 256 :
    --}}}
    INT frame, command :
    WHILE TRUE
      SEQ
        frame := 0
        from.master ? command
        WHILE command = c.guidance.run
          --{{{  run simulation
          REAL32 range, time:
          INT16 centroid.x, centroid.y :
          [3]REAL32 seeker.xyz, target.xyz :
          SEQ
            from.master ? range ; time ;
                          centroid.x ; centroid.y ;
                          seeker.xyz ; target.xyz
            --frame := frame + 1
            IF
              IF i = 0 FOR SIZE(begin.divert)
                (range  >= begin.divert[i]) AND (range <= end.divert[i])
                  --{{{  divert
                  INT shift.x, shift.y :
                  SEQ
                    shift.x := x.center - ((INT centroid.x) << 2)
                    shift.y := y.center - ((INT centroid.y) << 2)
                    IF
                      shift.x > max.shift.x
                        to.master ! (INT16 max.shift.x)
                      shift.x < min.shift.x
                        to.master ! (INT16 min.shift.x)
                      TRUE
                        to.master ! (INT16 shift.x)

                    IF
                      shift.y > max.shift.y
                        to.master ! (INT16 max.shift.y)
                      shift.y < min.shift.y
                        to.master ! (INT16 min.shift.y)
                      TRUE
                        to.master ! (INT16 shift.y)
```

```
                --}}}
              TRUE
                to.master ! 0 (INT16); 0 (INT16)
            from.master ? command

          --}}}
:
--}}}
CHAN OF ANY from.master, to.master :
PAR
  test.setup (from.master, to.master)
  --{{{
  BYTE length :
  [255]INT command.line :
  INT guidance.mode :
  REAL32 last.time :
  INT16 shift.x, shift.y :
  SEQ
    guidance.mode := gm.none
    WHILE TRUE
      SEQ
        fromSP ? length::command.line
        CASE command.line[0]
          --{{{  set mode
          c.guidance.set.mode
            guidance.mode := command.line[1]
          --}}}
          --{{{  initialize
          c.guidance.initialize
            SEQ
              last.time := -1.0 (REAL32)
              shift.x := 0 (INT16)
              shift.y := 0 (INT16)
              CASE guidance.mode
                gm.internal
                  from.master ! c.guidance.initialize
                gm.external
                  PAR
                    toXBar ! c.guidance.initialize
                    from.master ! c.guidance.initialize
                ELSE
                  SKIP
          --}}}
          --{{{  run
          c.guidance.run
            FPA          IS command.line[1] :
            range        IS command.line[2] :
            time         IS command.line[3] :
            centroid.x   IS command.line[4] :
            centroid.y   IS command.line[5] :
            seeker.int   IS [command.line FROM 6 FOR 3] :
            target.int   IS [command.line FROM 9 FOR 3] :
            [3]REAL32 seeker.xyz RETYPES seeker.int :
            [3]REAL32 target.xyz RETYPES target.int :
            REAL32 r.time RETYPES time :
            SEQ
              --{{{  convert positions to kilometers
              SEQ
                SEQ i = 0 FOR 3
                  SEQ
                    seeker.xyz[i] := seeker.xyz[i] / 1000.0 (REAL32)
                    target.xyz[i] := target.xyz[i] / 1000.0 (REAL32)
              --}}}
              CASE guidance.mode
```

```
                    gm.internal
                      --{{{  from test proc
                      SEQ
                        from.master ! c.guidance.run;
                                      range; time;  INT16  centroid.x  ;
INT16 centroid.y ;
                                    seeker.xyz ; target.xyz
                        to.master ? shift.x; shift.y
                         toSP !  cc.shift.image;  -(INT  shift.x);  -(INT
shift.y)
                      --}}}
                    gm.external
                      --{{{  from PFP
                      IF
                        ABS (r.time - last.time) >= min.time
                          SEQ
                            last.time := r.time
                            toXBar   ! c.guidance.run ;
                                       range; time  ;  INT16  centroid.x  ;
INT16 centroid.y ;
                                       seeker.xyz ; target.xyz
                            fromXBar ? shift.x; shift.y
                            toSP ! cc.shift.image; -(INT shift.x); -(INT
shift.y)
                        TRUE
                          --{{{  from test proc
                          SEQ
                            from.master ! c.guidance.run;
                                          range; time; INT16 centroid.x ;
INT16 centroid.y ;
                                          seeker.xyz ; target.xyz
                            to.master ? shift.x; shift.y
                            toSP ! cc.shift.image; -(INT shift.x); -(INT
shift.y)
                          --}}}
                      --}}}
                    ELSE
                      SKIP
              --}}}
    --}}}
:
```

6.2.1.15. PROC HostSeeker "hostseek.occ"

```
PROC HostSeeker (CHAN OF ANY fromSeeker, toSeeker )
  --{{{  libraries
  #INCLUDE "s_header.inc"
  #INCLUDE "hostio.inc"
  #USE "hostio.lib"
  --}}}
  --{{{  link definitions
  VAL  link0out   IS  0 :
  VAL  link1out   IS  1 :
  VAL  link2out   IS  2 :
  VAL  link3out   IS  3 :
  VAL  link0in    IS  4 :
  VAL  link1in    IS  5 :
  VAL  link2in    IS  6 :
  VAL  link3in    IS  7 :
  --}}}
  --{{{  channels
  CHAN OF SP fs, ts :
  PLACE fs AT link0in  :
  PLACE ts AT link0out :
  --}}}
  --{{{  constants
  VAL esc IS 27 :
  VAL max.screen.width  IS 640 :
  VAL max.screen.height IS 480 :
  --}}}
  --{{{  utility procs
  --{{{  goto.xy
  PROC goto.xy (CHAN OF SP fs, ts, VAL INT x, y)
    VAL esc IS 27 (BYTE) :
    SEQ
      so.write.string (fs, ts, [esc, '['])
      so.write.int (fs, ts, y+1, 0)
      so.write.char (fs, ts, ';')
      so.write.int (fs, ts, x+1, 0)
      so.write.char (fs, ts, 'H')
  :
  --}}}
  --{{{  clear.eos
  PROC clear.eos (CHAN OF SP fs, ts)
    VAL esc IS 27 (BYTE) :
    SEQ
      so.write.string (fs, ts, [esc, '[', 'J'])
  :
  --}}}
  --}}}
  --{{{  SC Encode
  #USE "gif.c8h"
  --{{{F gif.occ
  --:::F gif.OCC
  --}}}
  --}}}
  --{{{  SC loader
  #USE "loader.c8h"
  --{{{F loader.occ
  --:::F LOADER.OCC
  --}}}
  --}}}
  --{{{  SC runSeeker
  #USE "runseekr.c8h"
  --{{{F runSeekr.occ
```

```
  --:::F RUNSEEKR.OCC
  --}}}
  --}}}
  --{{{  spectrum
  PROC spectrum ( [256]INT palette )

    PROC set.colour (VAL INT index, r, g, b)
      SEQ
        palette[index] := (r << 18) \/ ((g << 10) \/ (b << 2))
    :
    SEQ
      SEQ i = 0 FOR 64                          -- blue to red scale for 1
to 63
        set.colour( i, i, 0, 31-(i>>1) )
      SEQ i = 0 FOR 64                          -- adding green and blue
        set.colour( 64+i, 63, i, i )            --    for 64 to 127

      SEQ i = 128 FOR 128                       -- green for 128-255
        set.colour( i, 0, 63, 0 )
      set.colour( 0, 0, 0, 0 )                  -- black for 0
      set.colour( 128, 30, 30, 30 )             -- grey for 128
  :
  --}}}
  --{{{  main menu variables
  BOOL dont.exit   :
  BYTE key, result :
  --}}}
  SEQ
    dont.exit := TRUE
    WHILE dont.exit
      SEQ
        --{{{  Main Menu
        goto.xy( fs, ts, 0, 0 )
        clear.eos( fs, ts )
        so.write.string( fs, ts, "Main Seeker Menu*c*n" )
        so.write.string( fs, ts, "----------------*c*n*n" )
        so.write.string( fs, ts, "*'Esc*' key to exit program.*c*n*n" )
        so.write.string( fs, ts, " (c)  capture screen image*c*n" )
        so.write.string( fs, ts, " (l)  load new data*c*n" )
        so.write.string( fs, ts, " (r)  run seeker*c*n" )
        so.write.nl( fs, ts )

        so.getkey( fs, ts, key, result )
        --}}}
        CASE key
          --{{{  c - capture screen
          'C', 'c'
            [max.screen.width]BYTE screen.buffer :
            [max.screen.height][max.screen.width]INT screen :
            [256]INT palette :
            INT screen.height, screen.width :
            SEQ
              --{{{  header
              goto.xy( fs, ts, 0, 0 )
              clear.eos( fs, ts )
              so.write.string( fs, ts, "Capture Screen Image*c*n" )
              so.write.string( fs, ts, "--------------------*c*n*n" )
              --}}}
              --{{{  get image
              spectrum(palette)

              --{{{  track display
                so.write.string( fs, ts, "Capturing track display
image...*c*n" )
```

```
                toSeeker ! 2 (BYTE); c.read.graphics ; track.display
                fromSeeker ? screen.height
                SEQ i = 0 FOR screen.height
                  SEQ
                    so.write.int (fs, ts, i, 0)
                    so.write.char (fs,ts, '*c')
                    fromSeeker ? screen.width::screen.buffer
                    SEQ j = 0 FOR screen.width
                      screen[i][j] := (INT screen.buffer[j])
                so.write.nl (fs, ts)
                --}}}
                --{{{  image display 0
                  so.write.string( fs,   ts,   "Capturing   image   display
0...*c*n" )
                toSeeker ! 2 (BYTE); c.read.graphics ; image.display.0
                fromSeeker ? screen.height
                SEQ i = 0 FOR screen.height
                  SEQ
                    so.write.int (fs, ts, i, 0)
                    so.write.char (fs,ts, '*c')
                    fromSeeker ? screen.width::screen.buffer
                    SEQ j = 0 FOR screen.width
                          screen[i][j] :=     screen[i][j]     \/     (INT
screen.buffer[j])
                so.write.nl (fs, ts)
                --}}}
                --{{{  image display 1
                  so.write.string( fs,   ts,   "Capturing   image   display
1...*c*n" )
                toSeeker ! 2 (BYTE); c.read.graphics ; image.display.1
                fromSeeker ? screen.height
                SEQ i = 0 FOR screen.height
                  SEQ
                    so.write.int (fs, ts, i, 0)
                    so.write.char (fs,ts, '*c')
                    fromSeeker ? screen.width::screen.buffer
                    SEQ j = 0 FOR screen.width
                          screen[i][j] :=     screen[i][j]     \/     (INT
screen.buffer[j])
                so.write.nl (fs, ts)
                --}}}
                --}}}
                --{{{  run gif encoder
                INT32 streamid :
                BYTE result :
                SEQ
                  so.open (fs, ts, "seeker.gif", spt.binary, spm.output,
                           streamid, result)
                  Encode (fs, ts, streamid, screen.width, screen.height,
                          8, palette, screen)
                --}}}
          --}}}
          --{{{  l - load new data
          'L', 'l'
            loader (fs, ts, fromSeeker, toSeeker)
          --}}}
          --{{{  r - run seeker
          'R', 'r'
            runSeeker (fs, ts, fromSeeker, toSeeker)
          --}}}
          --{{{  escape
          (BYTE esc)
            dont.exit := FALSE
          --}}}
```

```
          --{{{  else skip
          ELSE
            SKIP
          --}}}
    so.exit( fs, ts, 0 (INT32) )
:
```

6.2.1.16. PROC HostStub "hoststub.occ"

```
PROC HostStub (CHAN OF ANY from.emulator, to.emulator)
  SKIP
:
```

6.2.1.17. PROC ImageDisplay "imagedis.occ"

```
PROC ImageDisplay ( CHAN OF ANY Image0, Image1,
                                fromPrev, toPrev, fromNext, toNext,
                    VAL INT position, FRAMES )

  --{{{  libraries
  #INCLUDE  "crtc.inc"
  #INCLUDE  "g_header.inc"
  #INCLUDE  "s_header.inc"

  #USE  "convert.lib"
  #USE  "graphics.lib"
  --}}}
  --{{{  place system variables
  [(20*65536)+1280] BYTE screen.map :
  PLACE screen.map AT screen.int.address :

  INT DisplayStart :
  PLACE DisplayStart AT DisplayStart.address :

  INT EventMode :
  PLACE EventMode AT EventMode.address :

  INT SysReady :
  PLACE SysReady AT (#00080000  >< (MOSTNEG INT)) >> 2 :

  INT Ready :
  PLACE Ready AT Ready.address :
  --}}}
  --{{{  FrameReceiver
  PROC FrameReceiver ( CHAN OF ANY in0, in1,
                       [128][1280] BYTE frame,
                       VAL INT count0, count1 )

    --{{{  MOVE
    PROC MOVE( [][] BYTE source, VAL INT s.x, s.y,
               [][] BYTE dest,   VAL INT d.x, d.y, l.x, l.y )

      SEQ i = 0 FOR l.y
        [dest[d.y+i] FROM d.x FOR l.x] := [source[s.y+i] FROM s.x FOR
l.x]
    :
    --}}}
    [128][40][32] BYTE s RETYPES frame :
    [2][8][16] BYTE buffer0 :
    PLACE buffer0 IN WORKSPACE :
    [2][8][16] BYTE buffer1 :
    PLACE buffer1 IN WORKSPACE :
    SEQ
      --{{{  get first lines
      PAR
        in0 ? buffer0[0]
        in1 ? buffer1[0]
      --}}}
      --{{{  get lines and place on display
      SEQ i = 0 FOR 127
        VAL input   IS (i+1)\/1 :
        VAL display IS i\/1 :
        s1 IS s[i] :
        b0 IS buffer0[ display ] :
        b1 IS buffer1[ display ] :
        PAR
```

```
          in0 ? buffer0[input]
          --{{{  place on display
          SEQ
            MOVE2D( b0, 0, 0, si,  0,  0, 16, 8 )
            MOVE2D( b0, 0, 0, si,  0, 20, 16, 8 )
            MOVE2D( b1, 0, 0, si, 16,  0, 16, 8 )
            MOVE2D( b1, 0, 0, si, 16, 20, 16, 8 )
          --}}}
          in1 ? buffer1[input]
      --}}}
      --{{{  place last lines on display
      si IS s[127] :
      b0 IS buffer0[ 1 ] :
      b1 IS buffer1[ 1 ] :
      SEQ
        MOVE2D( b0, 0, 0, si,  0,  0, 16, 8 )
        MOVE2D( b0, 0, 0, si,  0, 20, 16, 8 )
        MOVE2D( b1, 0, 0, si, 16,  0, 16, 8 )
        MOVE2D( b1, 0, 0, si, 16, 20, 16, 8 )
      --}}}
:
--}}}
--{{{  place Event channel
CHAN OF ANY Event :
PLACE Event AT 8 :
--}}}
--{{{  set up multiple screens
VAL screen.offset IS  [ #00000, #50000, #A0000, #F0000 ] :

VAL screen.address IS [ #00000, #14000, #28000, #3C000 ] :
--}}}
--{{{  EventProc
PROC EventProc ( CHAN OF ANY Event, in )

  INT synch, address :
  WHILE TRUE
    SEQ
      in ? address
      Ready := 1
      Event ? synch
      DisplayStart := address
      Ready := 0
:
--}}}
--{{{  Buffer
PROC Buffer ( CHAN OF ANY in, out )

  INT temp :
  SEQ
    WHILE TRUE
      SEQ
        in ? temp
        out ! temp
:
--}}}
--{{{  SetNewScreen
PROC SetNewScreen ( CHAN OF ANY in, VAL INT screen.address )

  INT synch :
  SEQ
    Ready := 1
    in ? synch
    DisplayStart := screen.address
    SEQ i = 0 FOR 100
```

```
        SKIP
      Ready := 0
  :
  --}}}
  --{{{  constants
  VAL  screen.width   IS  640 :
  VAL  screen.height  IS  480 :
  VAL  screen.size    IS  screen.width * screen.height :
  --}}}
  --{{{  variables
  CHAN OF ANY synch, synch1 :
  INT temp, load :
  [w.length] INT window :

  INT start.time, end.time :
  TIMER clock :

  BYTE length :
  [256]INT message :
  --{{{  x.offset
  VAL offset IS [ 32, 352 ] :
  VAL x.offset IS offset[ position ] :
  --}}}
  --}}}
  SEQ
    --{{{  initialize
    set.B408( 0, 0, 0, 0, 0 )
    window := [ 0, 640*480, 640, 0, 0, 640, 480, 255, 0, 0, 0 ]

    [] INT int.screen RETYPES screen.map :
    SEQ i = 0 FOR (20*65536)/4
      int.screen[i] := 0

    set.B408( 0, 0, 0, 1, 0 )

    load := 0
    --}}}
    PRI PAR
      PAR
        EventProc( Event, synch1  )
        Buffer( synch, synch1 )
        --{{{  read display info
        command IS message[0] :
        who     IS message[1] :
        WHILE TRUE
          SEQ
            fromPrev ? length::message
            IF
              command = c.read.graphics
                CASE who
                  image.display.0  -- this processor
                    --{{{
                      screen IS  [screen.map  FROM  DisplayStart  FOR
screen.size ] :
                    [screen.height][screen.width] BYTE s2 RETYPES screen
:
                    SEQ
                      toPrev ! screen.height
                      SEQ i = 0 FOR screen.height
                        toPrev ! screen.width::s2[i]
                    --}}}
                  ELSE
                    --{{{  image.display 1
                    INT number.of.transfers, bufLength :
```

```
                    [maxGraphicBuffer]BYTE graphicsBuffer :
                    SEQ
                      message[1] := message[1] - 1    -- decrement and
send on
                      toNext   ! length::message
                      fromNext ? number.of.transfers
                      toPrev   ! number.of.transfers
                      SEQ i = 0 FOR number.of.transfers
                        SEQ
                          fromNext ? bufLength::graphicsBuffer
                          toPrev     ! bufLength::graphicsBuffer
                    --}}}
              TRUE
                SKIP
        --}}}
      --{{{  get display
      SEQ
        WHILE TRUE
          VAL start IS   screen.offset[load] + ((112 * screen.width) +
x.offset) :
          VAL size  IS  256 * screen.width :
          screen1 IS [screen.map FROM start FOR size ] :
          [128][screen.width*2] BYTE screen RETYPES screen1 :
          SEQ
            FrameReceiver( Image0, Image1, screen, 16, 4 )
            synch ! screen.address[load]
            load := (load + 1) /\ 3
      --}}}
:
```

6.2.1.18. PROC Loader (used by Host) "loader.occ"

```
--{{{  libraries
#INCLUDE "s_header.inc"
#INCLUDE "hostio.inc"
#USE "hostio.lib"
--}}}
PROC loader (CHAN OF SP fs, ts, CHAN OF ANY from.seeker, to.seeker)

  --{{{  constants
  VAL esc IS 27 :

  VAL c.cal.frames IS 2 :
  VAL c.background IS 0 :
  VAL c.foreground IS 1 :
  VAL c.row.data IS 44 :

  VAL max.rows    IS 128 :
  VAL max.cols    IS 128 :
  VAL max.t.cols IS 512 :
  VAL max.fpa.frames IS 200 :
  VAL max.buff.size  IS 128 :
  VAL sub.pixel.x IS 4 :
  VAL sub.pixel.y IS 4 :


  --}}}
  --{{{  global variables
  INT num.frames, num.sim.frames :
  BOOL fpa.values, sim.values :

  [max.fpa.frames]REAL32 frame.rate, fpa.time, fpa.range :
  [6][max.fpa.frames]REAL32 fpa.position :

  [max.sim.frames]REAL32 sim.time, sim.range :
  [6][max.sim.frames]REAL32 sim.position :

  --}}}
  --{{{  channels
  VAL link.in.3  IS 7 :
  VAL link.out.3 IS 3 :

  CHAN OF ANY from.vax, to.vax :
  PLACE from.vax AT link.in.3  :
  PLACE to.vax   AT link.out.3 :
  --}}}
  --{{{  utility procs
  --{{{  goto.xy
  PROC goto.xy (CHAN OF SP fs, ts, VAL INT x, y)
    VAL esc IS 27 (BYTE) :
    SEQ
      so.write.string (fs, ts, [esc, '['])
      so.write.int (fs, ts, y+1, 0)
      so.write.char (fs, ts, ';')
      so.write.int (fs, ts, x+1, 0)
      so.write.char (fs, ts, 'H')
  :
  --}}}
  --{{{  clear.eos
  PROC clear.eos (CHAN OF SP fs, ts)
    VAL esc IS 27 (BYTE) :
    SEQ
      so.write.string (fs, ts, [esc, '[', 'J'])
```

```
:
--}}}
--}}}
--{{{  procs
--{{{  proc get.INT
PROC get.INT (CHAN OF ANY in, out, INT value)
  SEQ
    out ! 4
    in ? value
:
--}}}
--{{{  proc get.BYTE
PROC get.BYTE (CHAN OF ANY in, out, BYTE value)
  SEQ
    out ! 1
    in ? value
:
--}}}
--{{{  proc get.REAL32
PROC get.REAL32 (CHAN OF ANY in, out, REAL32 value)
  SEQ
    out ! 4
    in ? value
:
--}}}
--{{{  proc get.REAL32.vector
PROC get.REAL32.vector (CHAN OF ANY in, out, []REAL32 value)
  SEQ
    out ! ((SIZE(value)) * 4)
    in ? value
:
--}}}
--{{{  proc vax.filter
PROC vax.filter (CHAN OF ANY from.filter, to.filter, from.vax,
                 VAL BOOL dataFromTape)
  VAL buffer.size IS 16384 :
  VAL max.record  IS 4097 :
  BOOL continue, waiting.flag, full :
  INT need, count, start, end :
  BYTE count.byte :
  [buffer.size]BYTE buffer :
  SEQ
    continue := TRUE
    waiting.flag := FALSE
    start := 0
    end := 0
    full := FALSE
    IF
      dataFromTape
        --{{{  then use sophisticated buffer
        WHILE continue
          PRI ALT
            to.filter ? need
              --{{{
              IF
                need = 0
                  continue := FALSE
                TRUE
                  --{{{
                  IF
                    (end - start) >= need
                      SEQ
                        from.filter ! [buffer FROM start FOR need]
                        start := start + need
```

```
                    IF
                      start = end
                        SEQ
                          start := 0
                          end := 0
                      TRUE
                        SKIP
                TRUE
                  waiting.flag := TRUE

            --}}}
        --}}}
      (NOT full) & from.vax ? count.byte
        --{{{
        SEQ
          IF
            (count.byte <> (BYTE #5E))
              SEQ
                --{{{  get record
                count := (INT count.byte) /\ #0F
                SEQ i = 0 FOR 3
                  SEQ
                    from.vax ? count.byte
                    count := (count * 10) + ((INT count.byte) /\
#0F)
                count := count - 4
                from.vax ? [buffer FROM end FOR count]
                end := end + count
                IF
                  waiting.flag AND ((end - start) >= need)
                    --{{{
                    SEQ
                      from.filter ! [buffer FROM start FOR need]
                      start := start + need
                      IF
                        start = end
                          SEQ
                            start := 0
                            end := 0
                        TRUE
                          SKIP
                      waiting.flag := FALSE

                    --}}}
                  TRUE
                    SKIP
              --}}}
            TRUE
              SKIP
          full := end > (buffer.size - max.record)
        --}}}
      (waiting.flag AND full) & SKIP
        --{{{
        [max.record]BYTE temp :
        INT length :
        SEQ
          length := end - start
          [temp FROM 0 FOR length] := [buffer FROM start FOR
length]
          [buffer FROM 0 FOR length] := [temp FROM 0 FOR length]
          start := 0
          end := length
          full := FALSE
        --}}}
```

```
        --}}}
      TRUE
        --{{{
        WHILE continue
          SEQ
            to.filter ? need
            IF
              need = 0
                continue := FALSE
              TRUE
                SEQ
                  from.vax ? [buffer FROM 0 FOR need]
                  from.filter ! [buffer FROM 0 FOR need]
        --}}}
:
--}}}
--{{{  PROC fixVaxFloat
PROC fixVaxFloat (VAL REAL32 bad, REAL32 good)
  SEQ
    VAL [4]BYTE twiddle.dee RETYPES bad :
    [4]BYTE twiddle.dum RETYPES good :
    SEQ
      twiddle.dum[0] := twiddle.dee[2]
      twiddle.dum[1] := twiddle.dee[3]
      twiddle.dum[2] := twiddle.dee[0]
      twiddle.dum[3] := twiddle.dee[1]
      --twiddle.dum[3] := BYTE ((INT twiddle.dum[3]) + 127)
      INT16 dec.exp RETYPES [twiddle.dum FROM 2 FOR 2] :
      INT16 ieee.exp :
      VAL  zero IS  0 (INT16) :
      SEQ
        ieee.exp := dec.exp - 256 (INT16)
        IF
          (ieee.exp < zero) AND (dec.exp > zero)
            --{{{  fix very small number
            SEQ
              dec.exp := zero
            --}}}
          TRUE
            dec.exp := ieee.exp
:
--}}}
--{{{  PROC clear.screen
PROC clear.screen(CHAN OF SP fs, ts)
  SEQ
    goto.xy(fs, ts, 0,0)
    clear.eos(fs, ts)
:
--}}}
--{{{  PROC checkFloat
PROC checkFloat(VAL REAL32 in, REAL32 out)
  SEQ
    IF
      ISNAN(in)
        out := 0.0 (REAL32)
      (ABS(in)) > 1.0E+20 (REAL32)
        out := 1.0E+20 (REAL32)
      TRUE
        out := in
:
--}}}

--{{{  merge.sim.and.fpa (CHAN)
PROC merge.sim.and.fpa (CHAN OF ANY to.seeker)
```

```
    --{{{  INT FUNCTION MIN (INT,INT)
    INT FUNCTION MIN (VAL INT a, b)
      INT return :
      VALOF
        IF
          a <= b
            return := a
          TRUE
            return := b
        RESULT return
    :
    --}}}
    SEQ
      --{{{  display some text
      so.write.string(fs, ts,
        "FPA and Sim values have all been loaded.*c*n")
      so.write.string(fs, ts,
        "Will now merge data sets and send to Seeker Emulator*c*n")
      so.write.string(fs, ts,
         "range, time,  frame  rate,  and  interceptor  and  target
coordinates.*c*n")
      --}}}
      --{{{  compare sim frames to fpa frames
      INT frames.sent, frames.to.send, sim.count :
      SEQ
        --{{{  send sim.frames
        sim.count := 0
        frames.sent := 0
        WHILE sim.time[sim.count] < fpa.time[0]
          SEQ
            --{{{  compute range values
            range IS sim.range[sim.count] :
            SEQ
              range := 0.0 (REAL32)
              SEQ i = 0 FOR 3
                VAL temp IS  (sim.position[i][sim.count] -
                              sim.position[i + 3][sim.count]) :
                range := range + (temp * temp)
              range := SQRT (range)
            --}}}
            sim.count := sim.count + 1
        WHILE frames.sent < sim.count
          SEQ
              frames.to.send :=    MIN(sim.count    -    frames.sent,
max.buff.size)
            SEQ i = 0 FOR 6
              to.seeker ! BYTE (4 + frames.to.send) ;
                                c.sim.position; i;     frames.sent     ;
frames.to.send ;
                               [sim.position[i] FROM    frames.sent    FOR
frames.to.send]
            to.seeker ! BYTE (3 + frames.to.send) ;
                        c.frame.time ; frames.sent ; frames.to.send ;
                        [sim.time FROM frames.sent FOR frames.to.send]
            to.seeker ! BYTE (3 + frames.to.send) ;
                        c.frame.range ; frames.sent ; frames.to.send ;
                        [sim.range FROM frames.sent FOR frames.to.send]
            frames.sent := frames.sent + frames.to.send
        --}}}
        --{{{  get position values for fpa frames
        INT current.sim :
        SEQ
          current.sim := sim.count - 1
          SEQ i = 0 FOR num.frames
```

```
            SEQ
              WHILE ABS(fpa.time[i] - sim.time[current.sim+1]) <
                    ABS(fpa.time[i] - sim.time[current.sim])
                current.sim := current.sim + 1
              SEQ j = 0 FOR 6
                fpa.position[j][i] := sim.position[j][current.sim]
          SEQ i = 0 FOR 6
            to.seeker ! BYTE (4 + num.frames) ;
                        c.sim.position; i; sim.count ; num.frames ;
                        [fpa.position[i] FROM 0 FOR num.frames]
          to.seeker ! BYTE (3 + num.frames) ;
                      c.frame.time ; sim.count ; num.frames ;
                      [fpa.time FROM 0 FOR num.frames]
          to.seeker ! BYTE (3 + num.frames) ;
                      c.frame.rate ; sim.count ; num.frames ;
                      [frame.rate FROM 0 FOR num.frames]
          to.seeker ! BYTE (3 + num.frames) ;
                      c.frame.range ; sim.count ; num.frames ;
                      [fpa.range FROM 0 FOR num.frames]
          to.seeker ! BYTE (3) ;
                       c.sim.start.frames ;  sim.count  ;  (sim.count  +
num.frames) - 1
        --}}}

      --}}}
  :
  --}}}
  --{{{  load.background (CHAN, CHAN)
  PROC load.background(CHAN OF ANY from.vax, to.seeker,
                       VAL BOOL dataFromTape, autoload)

    CHAN OF ANY from.filter, to.filter :
    PAR
      vax.filter(from.filter, to.filter, from.vax, dataFromTape)
      --{{{  local variables
      BYTE key, result :
      BOOL error :
      --}}}
      SEQ
        clear.screen (fs, ts)
        so.write.string (fs, ts, "Loading background data...*c*n")
        --{{{  get background frames
        INT nnrows, nncols :
        REAL32 diam, fnum, pxspcx,
               pxspcy, filfax, filfay :
        [max.frames]REAL32 framrt, range, time :
        BYTE cr :
        SEQ
          so.write.string(fs, ts, "*c*nBackground data...*c*n")
          --{{{  file header
          get.INT   (from.filter, to.filter, nnrows)
          get.BYTE  (from.filter, to.filter, cr)
          get.INT   (from.filter, to.filter, nncols)
          get.BYTE  (from.filter, to.filter, cr)
          get.REAL32   (from.filter, to.filter, diam)
          get.BYTE  (from.filter, to.filter, cr)
          get.REAL32   (from.filter, to.filter, fnum)
          get.BYTE  (from.filter, to.filter, cr)
          get.REAL32(from.filter, to.filter, pxspcx)
          get.BYTE  (from.filter, to.filter, cr)
          get.REAL32(from.filter, to.filter, pxspcy)
          get.BYTE  (from.filter, to.filter, cr)
          get.REAL32(from.filter, to.filter, filfax)
          get.BYTE  (from.filter, to.filter, cr)
```

```
          get.REAL32(from.filter, to.filter, filfay)
          get.BYTE  (from.filter, to.filter, cr)
          --}}}
          --{{{  qeff and dark current
          [max.cols]REAL32 qeff, dkcurr :
          SEQ
            so.write.string(fs, ts,  "Loading  and  sending  quantum
efficiency data.*c*n")
            SEQ i = 0 FOR nnrows
              SEQ
                get.REAL32.vector(from.filter, to.filter,
                                  [qeff FROM 0 FOR nncols])
                so.write.real32( fs, ts, qeff[0], 10, 3 )
                so.write.char( fs, ts, '*c' )
                get.BYTE  (from.filter, to.filter, cr)
                to.seeker ! BYTE (2 + nncols );
                            c.gain.row ; (nnrows - i) - 1 ; [qeff FROM 0
FOR nncols]

            so.write.string(fs, ts, "Loading and sending dark current
data.*c*n")
            SEQ i = 0 FOR nnrows
              SEQ
                get.REAL32.vector(from.filter, to.filter,
                                  [dkcurr FROM 0 FOR nncols])
                so.write.real32( fs, ts, dkcurr[0], 10, 3 )
                so.write.char( fs, ts, '*c' )
                get.BYTE  (from.filter, to.filter, cr)
                to.seeker ! BYTE (2 + nncols );
                            c.offset.row ; (nnrows - i) - 1 ; [dkcurr
FROM 0 FOR nncols]
          --}}}
          get.INT(from.filter, to.filter, num.frames)
          get.BYTE  (from.filter, to.filter, cr)
          so.write.string(fs, ts, "Loading noise data now.*c*n")
          so.write.string(fs, ts, "Number of frames: ")
          so.write.int(fs, ts, num.frames, 0)
          so.write.nl(fs, ts)
          --{{{  if not autoload allow for reduced frames
          IF
            autoload
              SKIP
            TRUE
              SEQ
                so.write.string(fs, ts, "Force Number of frames to: ")
                so.read.echo.int(fs, ts, num.frames, error)
                so.write.nl (fs, ts)
          --}}}
          SEQ i = 0 FOR num.frames
            --{{{
            INT framid :
            SEQ
              --{{{  frame header
              REAL32 temp :
              SEQ
                get.INT(from.filter, to.filter, framid)
                get.BYTE  (from.filter, to.filter, cr)

                get.REAL32(from.filter, to.filter, temp)
                get.BYTE  (from.filter, to.filter, cr)
                range[i] := temp

                get.REAL32(from.filter, to.filter, time[i])
                get.BYTE  (from.filter, to.filter, cr)
```

```
                get.REAL32(from.filter, to.filter, framrt[i])
                get.BYTE  (from.filter, to.filter, cr)
              --}}}
              --{{{  show some data values
              so.write.int(fs, ts, framid, 10)
              so.write.real32(fs, ts, range[i]  , 10, 3)
              so.write.real32(fs, ts, time[i]   , 10, 3)
              so.write.real32(fs, ts, framrt[i] , 10, 3)
              so.write.nl(fs, ts)
              --}}}
              SEQ j = 0 FOR nnrows
                [max.cols]REAL32 noise.data :
                SEQ
                  get.REAL32.vector(from.filter, to.filter,
                                    [noise.data FROM 0 FOR nncols])
                  get.BYTE  (from.filter, to.filter, cr)
                  to.seeker ! BYTE ( 3 + nncols) ;
                              c.background.row ; i ; (nnrows - j) - 1;
                              [noise.data FROM 0 FOR nncols]
            --}}}
        --}}}
        so.write.nl (fs, ts)
        so.write.string(fs, ts, "Finished loading background data.*c*n
")
        IF
          autoload
            SKIP
          TRUE
            --{{{  get anything extra
            BYTE key, result :
            SEQ
               so.write.string(fs, ts, "Hit  any  key  when  file  load
finished.*c*n ")
              so.getkey(fs, ts, key, result)
            --}}}
        to.filter ! 0  -- terminate filter
  :
  --}}}
  --{{{  load.target (CHAN, CHAN)
  PROC load.target (CHAN OF ANY from.vax, to.seeker,
                    VAL BOOL dataFromTape, autoload)
    CHAN OF ANY from.filter, to.filter :
    PAR
      vax.filter(from.filter, to.filter, from.vax, dataFromTape)
      --{{{  get target frames
      --{{{  local variables
      BYTE key, result :
      BOOL error :
      INT nsize, start.frame :
      BYTE cr :
      --}}}
      SEQ
        --{{{  display title
        clear.screen(fs, ts)
        so.write.string(fs, ts, "Loading Target data now...*c*n")
        --}}}
        --{{{  load file header
        SEQ
          get.INT   (from.filter, to.filter, num.frames)
          get.BYTE  (from.filter, to.filter, cr)
          get.INT   (from.filter, to.filter, nsize)
          get.BYTE  (from.filter, to.filter, cr)
          so.write.string(fs, ts, "Number of frames is ")
```

```
          so.write.int(fs, ts, num.frames, 0)
          so.write.nl(fs, ts)
        --}}}
        --{{{  get target frames
        --{{{  display a header
        IF
          autoload
            SKIP
          TRUE
            SEQ
              so.write.string(fs, ts, "Force Number of frames to: ")
              so.read.echo.int(fs, ts, num.frames, error)
              so.write.nl (fs, ts)

        so.write.string(fs, ts, "Target data...*c*n")
        so.write.string(fs, ts, " Frame ID     range                time
rate")
        so.write.nl(fs, ts)
        --}}}
        --{{{  acquire data
        VAL num.full.rows IS nsize / sub.pixel.y :
        VAL num.full.cols IS nsize / sub.pixel.x :
        SEQ frame = 0 FOR num.frames
          INT framid :
          SEQ
            --{{{  frame id
            get.INT   (from.filter, to.filter, framid)
            get.BYTE  (from.filter, to.filter, cr)
            so.write.int(fs, ts, framid, 4)
            --}}}
            --{{{  everything else
            REAL32 temp :
            SEQ
              --{{{  range
              REAL32 temp :
              SEQ
                get.REAL32(from.filter, to.filter, temp)
                get.BYTE  (from.filter, to.filter, cr)
                fpa.range[frame] := temp
                so.write.string(fs, ts, " ")
                so.write.real32(fs, ts, fpa.range[frame], 10, 3)
              --}}}
              --{{{  time
              get.REAL32(from.filter, to.filter, fpa.time[frame])
              get.BYTE  (from.filter, to.filter, cr)
              so.write.string(fs, ts, " ")
              so.write.real32(fs, ts, fpa.time[frame], 10, 3)
              --}}}
              --{{{  framrt
              get.REAL32(from.filter, to.filter, frame.rate[frame])
              get.BYTE  (from.filter, to.filter, cr)
              so.write.string(fs, ts, " ")
              so.write.real32(fs, ts, frame.rate[frame], 10, 1)
              --}}}
              so.write.nl(fs, ts)
              SEQ i = 0 FOR nsize
                [max.t.cols]REAL32 target.data :
                [sub.pixel.x][max.cols]REAL32 arranged.data :
                SEQ
                  --{{{  get target from vax
                  get.REAL32.vector(from.filter, to.filter,
                                    [target.data FROM 0 FOR nsize/2])
                  get.BYTE  (from.filter, to.filter, cr)
```

```
                  get.REAL32.vector(from.filter, to.filter,
                                    [target.data FROM   nsize/2   FOR
nsize/2])
                  get.BYTE  (from.filter, to.filter, cr)

                  so.write.int(fs, ts, i, 0)
                  so.write.string(fs, ts,"*c")
                  --}}}
                  --{{{  correct and format it
                  INT count :
                  SEQ
                    count := 0
                    SEQ j = 0 FOR num.full.cols
                      SEQ k = 0 FOR sub.pixel.x
                        SEQ
                          arranged.data[k][j] := target.data[count]
                          count := count + 1
                  --}}}
                  --{{{  send it to seeker
                   --VAL subpixel  IS    ((sub.pixel.y  -  1)  -  (i  \
sub.pixel.y)) * sub.pixel.x :
                    --VAL current.row   IS   (num.full.rows   -   (i   /
sub.pixel.y)) - 1 :
                  VAL subpixel IS  (i \ sub.pixel.y) * sub.pixel.x :
                  VAL current.row IS (i / sub.pixel.y) :
                  SEQ j = 0 FOR sub.pixel.x
                    SEQ
                      to.seeker ! BYTE (4 + num.full.cols) ;
                                  c.target.row ; frame ;
                                  subpixel + j;
                                  current.row ;
                                     [arranged.data[j] FROM    0    FOR
num.full.cols]
                  --}}}
              --}}}
              so.write.string(fs, ts, "*c")
          --}}}
          --{{{  send frame.rate, fpa.range, fpa.time
          to.seeker ! BYTE(3 + num.frames) ;
                      c.frame.time; 0; num.frames;  [fpa.time   FROM 0 FOR
num.frames]
          to.seeker ! BYTE(3 + num.frames) ;
                      c.frame.rate; 0; num.frames;  [frame.rate FROM 0 FOR
num.frames]
          to.seeker ! BYTE(3 + num.frames) ;
                      c.frame.range; 0; num.frames; [fpa.range FROM 0 FOR
num.frames]
          to.seeker ! BYTE(3); c.sim.start.frames; 0; (num.frames-1)
          --}}}
          --}}}
          --{{{  finish up
          INT dummy :
          SEQ
            so.write.nl(fs, ts)
            so.write.string(fs, ts, "Finished loading target data.*c*n ")
            IF
              autoload
                SKIP
              TRUE
                SEQ
                  so.write.string(fs, ts, "Hit any key when file transfer
ends.*c*n ")
                  so.getkey (fs, ts, key, result)
            to.filter ! 0
```

```
        --}}}
        --{{{  check if sim values have been loaded
        IF
          sim.values
            merge.sim.and.fpa (to.seeker)
          TRUE
            SKIP
        fpa.values := TRUE
        --}}}
      --}}}
:
--}}}
--{{{  load.positions (CHAN, CHAN)
PROC load.positions (CHAN OF ANY from.vax, to.seeker)
  SEQ
    --{{{  load sim values
    BOOL continue :
    SEQ
      clear.screen(fs, ts)
      continue := TRUE
      num.sim.frames := 0
      WHILE continue
        SEQ
          from.vax ? sim.time[num.sim.frames]
          SEQ i = 0 FOR 6
            from.vax ? sim.position[i][num.sim.frames]
          --{{{  display values
          so.write.real32 (fs, ts, sim.time[num.sim.frames], 10, 3)
          so.write.char(fs, ts, '*c')
          --}}}
          IF
            sim.time[num.sim.frames] < 0.0 (REAL32)
              continue := FALSE
            TRUE
              num.sim.frames := num.sim.frames + 1
    --}}}
    --{{{  check if fpa values have been loaded
    IF
      fpa.values
        merge.sim.and.fpa (to.seeker)
      TRUE
        SKIP
    sim.values := TRUE
    --}}}
:
--}}}
--}}}

SEQ
  --{{{  initialize
  fpa.values := FALSE
  sim.values := FALSE
  --}}}
  --{{{  menu
  BYTE key, result :
  BOOL dont.exit :
  SEQ
    dont.exit := TRUE
    WHILE dont.exit
      SEQ
        --{{{  Loader Menu
        goto.xy( fs, ts, 0, 0 )
        clear.eos( fs, ts )
        so.write.string( fs, ts, "Load New Seeker Data Menu*c*n" )
```

```
            so.write.string( fs, ts, "----------------*c*n*n" )
            so.write.string( fs, ts, "*'Esc*' key to exit program.*c*n*n"
)
            so.write.string( fs, ts, " (b)  background data*c*n" )
            so.write.string( fs, ts, " (t)  target data*c*n" )
            so.write.string( fs, ts, " (p)  position information*c*n" )

            so.getkey( fs, ts, key, result )
            --}}}
            CASE key
              --{{{  b - background data
              'B', 'b'
                load.background(from.vax, to.seeker, FALSE, FALSE)
              --}}}
              --{{{  t - target
              'T', 't'
                load.target(from.vax, to.seeker, TRUE, FALSE)
              --}}}
              --{{{  p - position info
              'P', 'p'
                load.positions(from.vax, to.seeker)
              --}}}
              --{{{  escape
              (BYTE esc)
                dont.exit := FALSE
              --}}}
              --{{{  else
              ELSE
                SKIP
              --}}}
    --}}}
:
```

6.2.1.19. PROC runSeeker "runseekr.occ"

```
--{{{  libraries
#INCLUDE "s_header.inc"
#INCLUDE "hostio.inc"
#USE "hostio.lib"
--}}}
PROC runSeeker (CHAN OF SP fs, ts, CHAN OF ANY fromSeeker, toSeeker)
  --{{{  utility procs
  --{{{  goto.xy
  PROC goto.xy (CHAN OF SP fs, ts, VAL INT x, y)
    VAL esc IS 27 (BYTE) :
    SEQ
      so.write.string (fs, ts, [esc, '['])
      so.write.int (fs, ts, y+1, 0)
      so.write.char (fs, ts, ';')
      so.write.int (fs, ts, x+1, 0)
      so.write.char (fs, ts, 'H')
  :
  --}}}
  --{{{  clear.eos
  PROC clear.eos (CHAN OF SP fs, ts)
    VAL esc IS 27 (BYTE) :
    SEQ
      so.write.string (fs, ts, [esc, '[', 'J'])
  :
  --}}}
  --{{{  so.get.extended.key
  PROC so.get.extended.key (CHAN OF SP fs, ts, INT extended.key)
    BYTE key, result :
    SEQ
      so.getkey( fs, ts, key, result)
      IF
        key =  0 (BYTE)
          SEQ
            so.getkey (fs, ts, key, result)
            extended.key := 256 + (INT key)
        TRUE
          extended.key := (INT key)
  :
  --}}}
  --{{{  get.real32
  PROC get.real32 ( CHAN OF SP fs, ts, REAL32 value )

    BOOL error :
    SEQ
      so.read.echo.real32( fs, ts, value, error )
      WHILE error
        SEQ
          so.write.string( fs, ts, "*c*nIllegal Real Number : " )
          so.read.echo.real32( fs, ts, value, error )
  :
  --}}}
  --{{{  get.int
  PROC get.int ( CHAN OF SP fs, ts, INT value )

    BOOL error :
    SEQ
      so.read.echo.int( fs, ts, value, error )
      WHILE error
        SEQ
          so.write.string( fs, ts, "*c*nIllegal Integer : " )
          so.read.echo.int( fs, ts, value, error )
```

```
  :
  --}}}
  --}}}
  --{{{  constants
  VAL esc IS 27 :
  --}}}
  --{{{  move.table
  VAL move.table  IS    [ [ 328,   0,   1 ],       -- up
                          [ 336,   0, -1 ],        -- down
                          [ 331,   1,   0 ],       -- left
                          [ 333, -1,   0 ],        -- right
                          [ 329, -1,   1 ],        -- up and right
                          [ 337, -1, -1 ],         -- down and right
                          [ 335,   1, -1 ],        -- down and left
                          [ 327,   1,   1 ],       -- up and left
                          [  56,   0,   4 ],       -- up
                          [  50,   0, -4 ],        -- down
                          [  52,   4,   0 ],       -- left
                          [  54, -4,   0 ],        -- right
                          [  57, -4,   4 ],        -- up and right
                          [  51, -4, -4 ],         -- down and right
                          [  49,   4, -4 ],        -- down and left
                          [  55,   4,   4 ] ] :    -- up and left
  --}}}
  --{{{  variables
  BYTE key, result :
  BOOL running :
  --}}}
  SEQ
    --{{{  initialize
    so.write.string( fs, ts, "Initialize Start Frame (y/n) ")
    so.getkey (fs, ts, key, result)
    CASE key
      'Y', 'y'
        toSeeker ! 3(BYTE); c.sim.start.frames; 0; 0
      ELSE
        SKIP

    toSeeker ! 1(BYTE); c.test.controller
    toSeeker ! 2(BYTE); c.global.scale; 0.5E-3(REAL32)
    toSeeker ! 6(BYTE); c.set.calibration; 2;
                          2000000.0(REAL32); 80000000.0(REAL32);    750;
30000

    running := TRUE
    --}}}
    WHILE running
      SEQ
        --{{{  Run Seeker Menu
        goto.xy( fs, ts, 0, 0 )
        clear.eos( fs, ts )
        so.write.string( fs, ts, "Run Seeker Menu*c*n" )
        so.write.string( fs, ts, "-----------------*c*n*n" )
         so.write.string( fs, ts, "*'Esc*' key to return to previous
menu.*c*n*n" )
        so.write.string( fs, ts, " ( )  single frame step*c*n" )
        so.write.string( fs, ts, " (a)  set to first fpa frame*c*n" )
        so.write.string( fs, ts, " (c)  continous mode*c*n" )
        so.write.string( fs, ts, " (f)  set frame number*c*n" )
        so.write.string( fs, ts, " (g)  initialize guidance*c*n" )
        so.write.string( fs, ts, " (r)  restart from calibration*c*n" )
        so.write.string( fs, ts, " (s)  set A/D gain*c*n" )
        so.write.string( fs, ts, " (t)  set test mode*c*n" )
        so.write.string( fs, ts, " (x)  set first and last frame*c*n" )
```

```
        so.write.string( fs, ts, " (z)  set calibration*c*n" )
        so.write.nl( fs, ts )

        so.getkey( fs, ts, key, result )
        --}}}
        CASE key
          --{{{  single frame
          ' '
            toSeeker ! 1(BYTE); c.run.single
          --}}}
          --{{{  continuous
          'C', 'c'
            INT key :
            SEQ
              --{{{  display continuous menu
              goto.xy (fs, ts, 0, 0)
              clear.eos (fs, ts)
              so.write.string( fs, ts, "Continuous Mode*c*n" )
              so.write.string( fs, ts, "---------------*c*n*n" )
               so.write.string( fs,  ts,  "*'Esc*'  key  to  return  to
previous menu.*c*n" )
               so.write.string( fs,  ts,  " Press  cursor  keys  to  shift
image*c*n*n" )
              so.write.string( fs, ts, " Home   |   PgUp*c*n*n" )
              so.write.string( fs, ts, " <---        --->*c*n*n" )
              so.write.string( fs, ts, " End    |   PgDn*c*n*n" )
               so.write.string( fs,  ts,  " Use  *'Shift*'  key  for  faster
movement." )
              --}}}
              toSeeker ! 1(BYTE);c.run.continuous
              so.get.extended.key (fs, ts, key)
              WHILE (key <> esc)
                SEQ
                  IF
                    IF i = 0 FOR SIZE move.table
                      key = move.table[i][0]
                         toSeeker !  cc.shift.image;  move.table[i][1];
move.table[i][2]
                    TRUE
                      SKIP
                  so.get.extended.key (fs, ts, key)
              toSeeker ! cc.exit
          --}}}
          --{{{  restart
          'R', 'r'
            toSeeker ! 1(BYTE); c.restart
          --}}}
          --{{{  start from first fpa image
          'A', 'a'
            toSeeker ! 4(BYTE); c.start.frame; 1; 0; 1
          --}}}
          --{{{  scale
          'S', 's'
            REAL32 scale :
            SEQ
              so.write.string( fs, ts, "*c*nEnter new scale: " )
              get.real32( fs, ts, scale )
              toSeeker! 2(BYTE); c.global.scale; scale
          --}}}
          --{{{  test mode
          'T', 't'
            INT key :
            SEQ
              toSeeker ! 1(BYTE); c.test.background
```

```
            toSeeker ! 1(BYTE); c.test.controller
          --}}}
          --{{{  frame
          'F', 'f'
            INT relative, frame :
            SEQ
              so.write.string (fs, ts, "*c*nChange frame (y/n)? ")
              so.getkey (fs, ts, key, result)
              CASE key
                'Y', 'y'
                  --{{{  get frame
                  SEQ
                     so.write.string (fs,  ts,  "*c*nStart  relative  to
first FPA frame (y/n)? ")
                     so.getkey (fs, ts, key, result)
                     CASE key
                       'Y', 'y'
                         relative := 1
                       ELSE
                         relative := 0
                     so.write.string( fs, ts, "*c*nStart Frame: " )
                     get.int( fs, ts, frame )
                  --}}}
                ELSE
                  relative := -1
              so.write.string (fs, ts, "*c*nDisplay fixed frame (y/n)?
")
              so.getkey (fs, ts, key, result)
              CASE key
                'Y', 'y'
                  toSeeker ! 4(BYTE); c.start.frame; relative; frame; 0
                ELSE
                  toSeeker ! 4(BYTE); c.start.frame; relative; frame; 1
          --}}}
          --{{{  set calibration
          'Z', 'z'
            REAL32 c0, c1 :
            INT sp0, sp1 :
            SEQ
              so.write.string( fs, ts, "*c*nLevel for First Calibration
on Seeker: " )
              get.real32( fs, ts, c0 )
              so.write.string( fs, ts, "*c*nLevel for Second Calibration
on Seeker: " )
              get.real32( fs, ts, c1 )
              so.write.string( fs, ts, "*c*nLevel for First Calibration
on SP: " )
              get.int( fs, ts, sp0 )
              so.write.string( fs, ts, "*c*nLevel for Second Calibration
on SP: " )
              get.int( fs, ts, sp1 )
              toSeeker ! 6(BYTE); c.set.calibration; 2; c0; c1; sp0; sp1

          --}}}
          --{{{  set first frame
          'X', 'x'
            INT first, last :
            SEQ
              so.write.string( fs, ts, "*c*nEnter value for first frame:
" )
              get.int( fs, ts, first )
              so.write.string( fs, ts, "*c*nEnter value for last frame:
" )
              get.int( fs, ts, last )
```

```
            toSeeker ! 3(BYTE); c.sim.start.frames; first; last
          --}}}
          --{{{  guidance initialize
          'G', 'g'
            BOOL not.valid.key, abort :
            SEQ
              --{{{  display guidance menu
              goto.xy (fs, ts, 0, 0)
              clear.eos (fs, ts)
              so.write.string( fs, ts, "Guidance Selection*c*n" )
              so.write.string( fs, ts, "-------------------*c*n*n" )
               so.write.string( fs,  ts,  "*'Esc*'  key  to  return  to
previous menu.*c*n*n" )
              so.write.string( fs, ts, " (x)  crossbar test*c*n" )
              so.write.string( fs, ts, " (i)  internal test*c*n" )
              so.write.string( fs, ts, " (n)  no guidance*c*n" )
              --}}}
              not.valid.key := TRUE
              WHILE not.valid.key
                SEQ
                  not.valid.key := FALSE
                  abort := FALSE
                  so.getkey (fs, ts, key, result)
                  CASE key
                    --{{{  x
                    'X', 'x'
                         toSeeker !    2(BYTE);    c.guidance.set.mode;
gm.external
                    --}}}
                    --{{{  i
                    'I', 'i'
                         toSeeker !    2(BYTE);    c.guidance.set.mode;
gm.internal
                    --}}}
                    --{{{  n
                    'N', 'n'
                      toSeeker ! 2(BYTE); c.guidance.set.mode; gm.none
                    --}}}
                    --{{{  escape
                    (BYTE esc)
                      abort := TRUE
                    --}}}
                    --{{{  else
                    ELSE
                      not.valid.key := TRUE
                    --}}}
                IF
                  abort
                    SKIP
                  TRUE
                    toSeeker ! 1(BYTE); c.guidance.initialize
          --}}}
          --{{{  escape
          (BYTE esc)
            running := FALSE
          --}}}
          --{{{  other
          ELSE
            SKIP
          --}}}
:
```

6.2.1.20. PROC SecondBuffer (Graphics Buffer) "secondbu.occ"

```
PROC SecondBuffer ( CHAN OF ANY in, fromNext, toPrev,
                    VAL INT position, shift )

  --{{{  constants
  --}}}
  --{{{  variables
  [2][64] INT input.buffer :
  INT count :
  --}}}
  --{{{  channels
  CHAN OF ANY synch, internal :
  --}}}
  --{{{  Receiver
  PROC Receiver ( CHAN OF ANY in, out, [2][64] INT buffer )

    INT i :
    SEQ
      i := 0
      WHILE TRUE
        SEQ
          in ? buffer[i]
          out ! i
          i := 1 - i
  :
  --}}}
  --{{{  Extractor
  PROC Extractor ( CHAN OF ANY internal, in, out,
                   VAL INT count )

    --{{{  variables
    [2][64][2] BYTE buffer :
    INT output :
    --}}}
    SEQ
      internal ? buffer[0]
      output := 0
      WHILE TRUE
        SEQ
          SEQ i = 0 FOR count
            SEQ
              PAR
                out ! buffer[output]
                in ? buffer[ 1-output ]
              output := 1 - output
          PAR
            out ! buffer[output]
            internal ? buffer[1-output]
          output := 1 - output
  :
  --}}}
  --{{{  Formatter
  PROC Formatter ( CHAN OF ANY synch, out,
                   [2][64] INT input.buffer )

    --{{{  variables
    [2][64][4] BYTE b.in RETYPES input.buffer :
    [64][2] BYTE buffer :
    [64*2] BYTE buffer1 RETYPES buffer :
    INT in.ptr :
    --}}}
    SEQ
```

```
      WHILE TRUE
        SEQ
          --{{{  form message in buffer
          SEQ
            synch ? in.ptr

            source IS input.buffer[in.ptr] :
            INT p :
            SEQ
              p := 0
              SEQ i = 0 FOR 64
                INT store :
                SEQ
                  store := source[i] >> shift
                  --{{{  check for zeroing store
                  IF
                    store = 0
                      IF
                        source[i] <> 0
                          store := 1
                        TRUE
                          store := 128
                    TRUE
                      SKIP
                  --}}}
                  buffer1[p] := BYTE store
                  buffer1[p+1] := BYTE store
                  p := p + 2
          --}}}
          out ! buffer
  :
  --}}}
  SEQ
    IF
      position < 8
        count := 7 - position
      TRUE
        count := 15 - position
    PRI PAR
      PAR
        Receiver( in, synch, input.buffer )
        Extractor( internal, fromNext, toPrev, count )
      Formatter( synch, internal, input.buffer )
:
```

6.2.1.21. PROC SP (Signal Processing) "sp.occ"

```
PROC SP ( CHAN OF ANY in, out, fromPrev, toPrev, fromNext, toNext,
          VAL INT position )

  #INCLUDE "s_header.inc"
  --{{{  constants
  VAL  packet.length    IS   8 :
  VAL  num.packets      IS   (128 * 8) / packet.length :

  VAL  fraction.bits    IS    8 :
  VAL  tolerance        IS    4 :
  --}}}
  --{{{  ProcessFrame
  PROC ProcessFrame ( CHAN OF ANY in, out,
                      VAL INT lower.threshold, upper.threshold,
                      [128][8] INT Gain, Offset,
                      INT  max.data, max.row, max.col )

    --{{{  constants
    VAL  l.thr   IS   lower.threshold << fraction.bits :
    VAL  u.thr   IS   upper.threshold << fraction.bits :
    --}}}
    --{{{  ProcessRow
    PROC ProcessRow ( [packet.length] INT data, gain, offset, VAL INT
row )

      SEQ i = 0 FOR packet.length
        INT value :
        SEQ
          value := (data[i] TIMES gain[i]) + offset[i]
          IF
            value < l.thr
              data[i] := 0
            value > u.thr
              data[i] := upper.threshold
            TRUE
              data[i] := value >> fraction.bits
          --{{{  find hot spot
          IF
            data[i] > max.data
              SEQ
                max.data := data[i]
                max.row  := row
                max.col  := i
            TRUE
              SKIP
          --}}}
    :
    --}}}
    --{{{  retype array to packet.length
    [num.packets][packet.length] INT  p.gain        RETYPES Gain       :
    [num.packets][packet.length] INT  p.offset      RETYPES Offset     :
    --}}}
    --{{{  variables
    INT in.ptr, out.ptr, process.ptr, temp :
    [3][packet.length] INT buffer :
    PLACE buffer IN WORKSPACE :
    --}}}
    SEQ
      --{{{  initialize
      in.ptr := 2
      process.ptr := 1
```

```
      out.ptr := 0

      max.data := -1
      max.row := 0
      max.col := 0
      --}}}
      --{{{  get first row
      in ? buffer[0]
      --}}}
      --{{{  get second row and process first row
      PRI PAR
        in ? buffer[1]
        ProcessRow( buffer[0], p.gain[1], p.offset[1], 0 )
      --}}}
      --{{{  do middle rows
      SEQ row = 1 FOR (num.packets-2)
        SEQ
          PRI PAR
            PAR
              in ? buffer[in.ptr]
              out ! buffer[out.ptr]
            ProcessRow( buffer[process.ptr], p.gain[row], p.offset[row],
row )
          temp := out.ptr
          out.ptr := process.ptr
          process.ptr := in.ptr
          in.ptr := temp
      --}}}
      --{{{  process last row
      VAL i IS num.packets - 1 :
      PRI PAR
        out ! buffer[out.ptr]
        ProcessRow( buffer[process.ptr],  p.gain[i], p.offset[i], i )
      --}}}
      --{{{  output last row
      out ! buffer[ process.ptr ]
      --}}}
      --{{{  determine actual max.col position
      max.col := (max.col << 4) + position
      --}}}
  :
  --}}}
  --{{{  CalibrateFrame
  PROC CalibrateFrame ( CHAN OF ANY in, out,
                        VAL INT c.frame, c.level,
                        [128][8] INT Gain, Offset )

    --{{{  global variables
    INT D0, D1, d0, delta.D, half.range :
    --}}}
    --{{{  ProcessRow
    PROC ProcessRow ( [packet.length] INT data, gain, offset )

      IF
        c.frame = 0
          gain := data
        TRUE
          SEQ i = 0 FOR packet.length
            INT previous.pixel, delta.p :
            SEQ
              previous.pixel := gain[i]
              delta.p := data[i] - previous.pixel
              IF
                delta.p < tolerance
```

```
                    --{{{  dead pixel
                    SEQ
                      gain[i] := 0
                        offset[i] :=   (previous.pixel   -   tolerance)   <<
fraction.bits
                    --}}}
                  TRUE
                    --{{{  normal pixel
                    SEQ
                      gain[i] := delta.D / delta.p
                      offset[i] := (d0 - (gain[i] TIMES previous.pixel)) +
half.range
                    --}}}
      :
      --}}}
      --{{{  retype array to packet.length
      [num.packets][packet.length] INT  p.gain        RETYPES Gain       :
      [num.packets][packet.length] INT  p.offset      RETYPES Offset     :
      --}}}
      --{{{  variables
      INT in.ptr, out.ptr, process.ptr, temp :
      [3][packet.length] INT buffer :
      PLACE buffer IN WORKSPACE :
      --}}}
      SEQ
        --{{{  set up calibration
        IF
          c.frame = 0
            D0 := c.level
          TRUE
            SEQ
              D1 := c.level
              delta.D := (D1 - D0) << fraction.bits
              d0 := D0 << fraction.bits
              half.range := 1 << (fraction.bits-1)
        --}}}
        in.ptr := 2
        process.ptr := 1
        out.ptr := 0
        --{{{  get first row
        in ? buffer[0]
        --}}}
        --{{{  get second row and process first row
        PRI PAR
          in ? buffer[1]
          ProcessRow( buffer[0], p.gain[1], p.offset[1] )
        --}}}
        --{{{  do middle rows
        SEQ row = 1 FOR (num.packets-2)
          SEQ
            PRI PAR
              PAR
                in ? buffer[in.ptr]
                out ! buffer[out.ptr]
              ProcessRow( buffer[process.ptr], p.gain[row], p.offset[row]
)
            temp := out.ptr
            out.ptr := process.ptr
            process.ptr := in.ptr
            in.ptr := temp
        --}}}
        --{{{  process last row
        VAL i IS num.packets - 1 :
        PRI PAR
```

```
        out ! buffer[out.ptr]
        ProcessRow( buffer[process.ptr],  p.gain[i], p.offset[i] )
      --}}}
      --{{{  output last row
      out ! buffer[ process.ptr ]
      --}}}
  :
  --}}}
  --{{{  variables
  --{{{  command variables
  BYTE length :
  [max.message] INT message :
  command IS message[0] :
  params IS [message FROM 1 FOR (max.message-1)] :
  --}}}

  [128][8] INT Gain, Offset :
  INT max.data, max.row, max.col :
  --}}}
  SEQ
    --{{{  initialize
    --}}}
    WHILE TRUE
      SEQ
        --{{{  get command and pass on
        fromPrev ? length::message
        IF
          position < 15
            toNext ! length::message
          TRUE
            SKIP
        --}}}
        --{{{  process command
        CASE command
          --{{{  c.sp.frame
          c.sp.frame
            IF
              params[0] < 0
                SEQ
                   ProcessFrame( in,  out,  params[2],  params[3],  Gain,
Offset,
                                 max.data, max.row, max.col )
                  --{{{  return hot spot
                  IF
                    position < 15
                      INT next.data, next.row, next.col :
                      SEQ
                        fromNext ? next.data; next.row; next.col
                        IF
                          next.data > max.data
                            toPrev ! next.data; next.row; next.col
                          TRUE
                            toPrev ! max.data; max.row; max.col
                    TRUE
                      toPrev ! max.data; max.row; max.col

                  --}}}
              TRUE
                 CalibrateFrame( in,  out,  params[0],  params[1],  Gain,
Offset )
          --}}}
        --}}}
:
```

6.2.1.22. PROC SPController "spcontro.occ"

```
PROC SPController ( CHAN OF ANY fromController, toController,
                                fromGuidance, toGuidance,
                                fromGraphics, toGraphics, fromSP, toSP )

  PAR
    --{{{  transfer shift command from guidance to controller
    INT command, shift.x, shift.y :
    SEQ
      WHILE TRUE
        SEQ
          fromGuidance ? command; shift.x; shift.y
          toController ! command; shift.x; shift.y
    --}}}
    --{{{  control SP and give guidance commands
    #INCLUDE "s_header.inc"
    --{{{  variables
    BYTE length :
    [max.message] INT message :
    command IS message[0] :
    params IS [message FROM 1 FOR (max.message-1)] :
    --}}}
    SEQ
      WHILE TRUE
        SEQ
          fromController ? length::message
          IF
            --{{{  Guidance commands
            (command >= 1280) AND (command < 1536)
              toGuidance ! length::message
            --}}}
            --{{{  Track Display commands
            (command >= 1536) AND (command < 1792)
              toGraphics ! length::message
            --}}}
            --{{{  c.sp.frame
            command = c.sp.frame
              SEQ
                toSP ! length::message
                IF
                  params[0] < 0
                    --{{{  regular frame
                    INT max.data, max.row, max.col :
                    SEQ
                      fromSP ? max.data; max.row; max.col
                        toGuidance !   12(BYTE);   c.guidance.run;   1;
params[4]; params[5];
                                                    max.col; max.row;  [params
FROM 8 FOR 6]
                      toGraphics ! 6(BYTE); c.display.info; 1; [params
FROM 4 FOR 4]
                    --}}}
                  TRUE
                    --{{{  calibration frame
                    SEQ
                      toGraphics ! 6(BYTE); c.display.info; 1;
                                            [params FROM 4 FOR 4]
                    --}}}
            --}}}
            --{{{  c.read.graphics
            command = c.read.graphics
              INT bufLength :
```

```
                INT number.of.transfers :
                [maxGraphicBuffer]BYTE graphicsBuffer :
                SEQ
                  toGraphics   ! length::message
                  fromGraphics ? number.of.transfers
                  toController ! number.of.transfers
                  SEQ i = 0 FOR number.of.transfers
                    SEQ
                      fromGraphics ? bufLength::graphicsBuffer
                      toController ! bufLength::graphicsBuffer
              --}}}
    --}}}
:
```

6.2.1.23. PROC Target "target.occ"

```
PROC Target ( CHAN OF ANY fromDown, toDown, fromUp, toUp,
              VAL INT column.position, row.position )

  #INCLUDE "s_header.inc"
  --{{{  constants
  VAL first.row  IS  row.position TIMES 16 :
  VAL next.first.row  IS  first.row + 16 :
  --}}}
  --{{{  SendFrame
  PROC SendFrame ( CHAN OF ANY in, out, [16][8] REAL32 target,
                   VAL INT shift.y, shift.x )

    --{{{  variables
    INT start.col, crossbar.offset :
    INT total.in, diff :
    INT output :

    [16][8] INT buffer :
    [2][8] INT b :
    --}}}
    SEQ
      --{{{  calculate values
      crossbar.offset := shift.x /\ 15
      start.col := shift.x >> 4
      IF
        column.position < crossbar.offset
          start.col := start.col + 1
        TRUE
          SKIP

      total.in := (7 - row.position) << 4
      --}}}
      --{{{  shift target data
      [16][8*4] BYTE b.buffer RETYPES buffer :
      [16][8*4] BYTE b.target RETYPES target :

      VAL start IS start.col << 2 :
      VAL length IS (8 - start.col) << 2 :
      SEQ
        IF
          start <> 32
            MOVE2D( b.target, start, 0, b.buffer, 0, 0, length, 16 )
          TRUE
            SKIP
        IF
          start.col <> 0
            MOVE2D( b.target, 0, 0, b.buffer, length, 0, start, 16 )
          TRUE
            SKIP
      --}}}
      --{{{  send target data
      diff := shift.y - (row.position << 4)
      IF
        row.position = 7
          --{{{  no one is above you
          SEQ
            IF
              diff < 1
                diff := 0
              TRUE
                SKIP
```

```
      SEQ i = diff FOR (16-diff)
        out ! buffer[i]
      SEQ i = 0 FOR diff
        out ! buffer[i]
  --}}}
diff < 0
  --{{{  start row is below
  SEQ
    PAR
      in ? b[0]
      SEQ i = 0 FOR 16
        out ! buffer[i]
    output := 0
    SEQ i = 0 FOR (total.in - 1)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
  --}}}
(diff - 16) <= 0
  --{{{  start row is in the middle
  SEQ
    PAR
      in ? b[0]
      SEQ i = diff FOR 16-diff
        out ! buffer[i]
    output := 0
    SEQ i = 0 FOR (total.in - 1)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
    SEQ i = 0 FOR diff
      out ! buffer[i]
  --}}}
TRUE
  --{{{  start row is in middle of processor further down
  SEQ
    in ? b[0]
    output := 0
    SEQ i = 0 FOR (127 - shift.y)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
    PAR
      in ? b[output]
      SEQ i = 0 FOR 16
        out ! buffer[i]
    SEQ i = 0 FOR ((total.in + shift.y) - 129)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
  --}}}
--}}}
```

```
  :
  --}}}
  --{{{  GetTargetRow
  PROC GetTargetRow ( [16][8] REAL32 Target, VAL INT row, [128] INT data
)

    [] REAL32 r.data RETYPES data :
    IF
      (row >= first.row) AND (row < next.first.row)
        SEQ i = 0 FOR 8
          Target[ row-first.row ][ i ] := r.data[ (i*16) +
column.position ]
      TRUE
        SKIP
  :
  --}}}
  --{{{  variables
  BYTE length :
  [max.message] INT message :
  command IS message[0] :
  params IS [message FROM 1 FOR (max.message-1)] :

  [max.frames][16][16][8] REAL32 Frame :
  --}}}
  SEQ
    --{{{  initialize last target frame
    SEQ r = 0 FOR 64
      SEQ c = 0 FOR 8
        SEQ o = 0 FOR 4
          INT column, row :
          REAL32 value :
          SEQ
            --{{{  determine col value
            column := (((c << 6) + o) + (column.position << 2)) + 1
            IF
              column < 256
                SKIP
              TRUE
                column := 513 - column
            --}}}
            --{{{  determine row value
            row := ((first.row << 2) + r) + 1
            IF
              row < 256
                SKIP
              TRUE
                row := 513 - row
            --}}}
            value  :=  REAL32 ROUND( (row * column)  )
            Frame[max.frames-1][((r/\3)<<2)+o][r>>2][c] := value
    --}}}
    WHILE TRUE
      SEQ
        --{{{  get command and send up
        fromDown ? length::message
        IF
          row.position < 7
            toUp ! length::message
          TRUE
            SKIP
        --}}}
        CASE command
          --{{{  c.set.target
          c.set.target
```

```
            SendFrame( fromUp, toDown, Frame[ params[0] ][ params[1] ],
                       params[2], params[3] )
          --}}}
          --{{{  c.target.row
          c.target.row
            GetTargetRow( Frame[ params[0] ][ params[1] ], params[2],
                          [params FROM 3 FOR 128] )
          --}}}
:
```

6.2.1.24. PROC TargetLead "targetle.occ"

```
PROC TargetLead ( CHAN OF ANY fromUp, toUp, toDown,
                              fromPrev, toPrev, fromNext, toNext,
                  VAL INT column.position )

  #INCLUDE "s_header.inc"
  --{{{  constants
  VAL row.position     IS  0 :
  VAL first.row        IS  0 :
  VAL next.first.row   IS 16 :
  --}}}
  --{{{  SendFrame
  PROC SendFrame ( CHAN OF ANY in, out, [16][8] REAL32 target,
                   VAL INT shift.y, shift.x )

    --{{{  variables
    INT start.col, crossbar.offset :
    INT total.in, diff :
    INT output :

    [16][8] INT buffer :
    [2][8] INT b :
    --}}}
    SEQ
      --{{{  calculate values
      crossbar.offset := shift.x /\ 15
      start.col := shift.x >> 4
      IF
        column.position < crossbar.offset
          start.col := start.col + 1
        TRUE
          SKIP

      total.in := (7 - row.position) << 4
      --}}}
      --{{{  shift target data
      [16][8*4] BYTE b.buffer RETYPES buffer :
      [16][8*4] BYTE b.target RETYPES target :

      VAL start IS start.col << 2 :
      VAL length IS (8 - start.col) << 2 :
      SEQ
        IF
          start <> 32
            MOVE2D( b.target, start, 0, b.buffer, 0, 0, length, 16 )
          TRUE
            SKIP
        IF
          start.col <> 0
            MOVE2D( b.target, 0, 0, b.buffer, length, 0, start, 16 )
          TRUE
            SKIP
      --}}}
      --{{{  send target data
      diff := shift.y - (row.position << 4)
      IF
        row.position = 7
          --{{{  no one is above you
          SEQ
            IF
              diff < 1
                diff := 0
```

```
        TRUE
          SKIP
      SEQ i = diff FOR (16-diff)
        out ! buffer[i]
      SEQ i = 0 FOR diff
        out ! buffer[i]
  --}}}
diff < 0
  --{{{  start row is below
  SEQ
    PAR
      in ? b[0]
      SEQ i = 0 FOR 16
        out ! buffer[i]
    output := 0
    SEQ i = 0 FOR (total.in - 1)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
  --}}}
(diff - 16) <= 0
  --{{{  start row is in the middle
  SEQ
    PAR
      in ? b[0]
      SEQ i = diff FOR 16-diff
        out ! buffer[i]
    output := 0
    SEQ i = 0 FOR (total.in - 1)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
    SEQ i = 0 FOR diff
      out ! buffer[i]
  --}}}
TRUE
  --{{{  start row is in middle of processor further down
  SEQ
    in ? b[0]
    output := 0
    SEQ i = 0 FOR (127 - shift.y)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
    PAR
      in ? b[output]
      SEQ i = 0 FOR 16
        out ! buffer[i]
    SEQ i = 0 FOR ((total.in + shift.y) - 129)
      SEQ
        PAR
          in ? b[1-output]
          out ! b[output]
        output := 1-output
    out ! b[output]
```

```
          --}}}
      --}}}
  :
  --}}}
  --{{{  GetTargetRow
  PROC GetTargetRow ( [16][8] REAL32 Target, VAL INT row, [128] INT data
)

    [] REAL32 r.data RETYPES data :
    IF
      (row >= first.row) AND (row < next.first.row)
        SEQ i = 0 FOR 8
          Target[ row-first.row ][ i ] := r.data[ (i*16) +
column.position ]
      TRUE
        SKIP
  :
  --}}}
  --{{{  variables
  BYTE length :
  [max.message] INT message :
  command IS message[0] :
  params IS [message FROM 1 FOR (max.message-1)] :

  [max.frames][16][16][8] REAL32 Frame :
  --}}}
  SEQ
    --{{{  initialize last target frame
    SEQ r = 0 FOR 64
      SEQ c = 0 FOR 8
        SEQ o = 0 FOR 4
          INT column, row :
          REAL32 value :
          SEQ
            --{{{  determine col value
            column := (((c << 6) + o) + (column.position << 2)) + 1
            IF
              column < 256
                SKIP
              TRUE
                column := 513 - column
            --}}}
            --{{{  determine row value
            row := ((first.row << 2) + r) + 1
            IF
              row < 256
                SKIP
              TRUE
                row := 513 - row
            --}}}
            value  :=  REAL32 ROUND( (row * column)  )
            Frame[max.frames-1][((r/\3)<<2)+o][r>>2][c] := value
    --}}}
    WHILE TRUE
      SEQ
        --{{{  get command and send up and accross
        fromPrev ? length::message
        PAR
          toUp ! length::message
          IF
            column.position < 15
              toNext ! length::message
            TRUE
              SKIP
```

```
        --}}}
        CASE command
          --{{{  c.set.target
          c.set.target
            SEQ
              SendFrame( fromUp, toDown, Frame[ params[0] ][ params[1]
],
                         params[2], params[3] )
              IF
                column.position = 15
                  toPrev ! 0 (BYTE)
                TRUE
                  BYTE synch :
                  SEQ
                    fromNext ? synch
                    toPrev ! synch
          --}}}
          --{{{  c.target.row
          c.target.row
            GetTargetRow( Frame[ params[0] ][ params[1] ], params[2],
                          [params FROM 3 FOR 128] )
          --}}}
:
```

6.2.1.25. PROC TrackDisplay "trackdis.occ"

```
PROC TrackDisplay ( CHAN OF ANY fromSP, toSP,
                                fromPrev, toPrev, fromNext, toNext )

  --{{{  libraries
  #INCLUDE  "s_header.inc"
  #INCLUDE  "g_header.inc"
  #INCLUDE  "crtc.inc"

  #USE  "convert.lib"
  #USE  "graphics.lib"
  --#USE  "extrio.lib"

  --}}}
  --{{{  display constants
  VAL  width           IS       640 :
  VAL  height          IS       480 :
  VAL  line.frequency  IS     60000 :
  VAL  frame.rate      IS        90 :
  VAL  pixel.clock     IS  64000000 :
  VAL  interlace       IS     FALSE :
  --}}}
  --{{{  fonts
  #INCLUDE "sys6.inc"
  --}}}
  --{{{  place system variables
  [(20*65536)+1280] BYTE screen.map :
  PLACE screen.map AT screen.int.address :

  INT DisplayStart :
  PLACE DisplayStart AT DisplayStart.address :

  INT EventMode :
  PLACE EventMode AT EventMode.address :

  INT SysReady :
  PLACE SysReady AT (#00080000  >< (MOSTNEG INT)) >> 2 :

  INT Ready :
  PLACE Ready AT Ready.address :
  --}}}
  --{{{  set up multiple screens
  VAL screen.offset IS  [ #00000, #50000, #A0000, #F0000 ] :

  VAL screen.address IS [ #00000, #14000, #28000, #3C000 ] :
  --}}}
  --{{{  place Event channel
  CHAN OF ANY Event :
  PLACE Event AT 8 :
  --}}}
  --{{{  constants
  VAL  screen.width   IS  640 :
  VAL  screen.height  IS  480 :
  VAL  screen.size    IS  screen.width * screen.height :

  VAL  char.width     IS   18 :
  VAL  char.height    IS   33 :
  --}}}
  --{{{  procs
  --{{{  spectrum
  PROC spectrum ( CHAN OF ANY out )
```

```
    SEQ
      SEQ i = 0 FOR 64                              -- blue to red scale for
1 to 63
        set.colour( out, 0, i, i, 0, 31-(i>>1) )
      SEQ i = 0 FOR 64                              -- adding green and blue
        set.colour( out, 0, 64+i, 63, i, i )        --    for 64 to 127

      SEQ i = 128 FOR 128                           -- green for 128-255
        set.colour( out, 0, i, 0, 63, 0 )
      set.colour( out, 0, 0, 0, 0, 0 )              -- black for 0
      set.colour( out, 0, 128, 30, 30, 30 )         -- grey for 128
  :
  --}}}
  --{{{  center.string
  PROC center.string ( [] INT window, [] BYTE screen,
                       VAL INT cx, sy, VAL [] BYTE string,
                       VAL [] INT font )

    INT width :
    SEQ
      string.width( font, string, width )
      window[ w.cursor.y ] := sy
      window[ w.cursor.x ] := cx - (width >> 1)
      write.string( window, screen, string, font )
  :
  --}}}
  --{{{  place.string
  PROC place.string ( [] INT window, [] BYTE screen,
                      VAL INT sx, sy, VAL [] BYTE string,
                      VAL [] INT font )

    SEQ
      window[ w.cursor.y ] := sy
      window[ w.cursor.x ] := sx
      write.string( window, screen, string, font )
  :
  --}}}
  --{{{  EventProc
  PROC EventProc ( CHAN OF ANY Event, in )

    INT synch, address :
    WHILE TRUE
      SEQ
        in ? address
        Ready := 1
        Event ? synch
        DisplayStart := address
        Ready := 0
  :
  --}}}
  --{{{  Buffer
  PROC Buffer ( CHAN OF ANY in, out )

    INT temp :
    SEQ
      WHILE TRUE
        SEQ
          in ? temp
          out ! temp
  :
  --}}}
  --{{{  set.text.window
  PROC set.text.window ( [] INT window,
                         VAL [] BYTE string1, string2, VAL [] INT font,
```

```
                        VAL INT start.x, start.y, size.y )

    SEQ
      string.width( font, string1, window[ w.start.x ] )
      string.width( font, string2, window[ w.size.x ] )
      window[ w.start.x ] := window[ w.start.x ] + start.x
      window[ w.size.x  ] := window[ w.size.x  ] + 1
      window[ w.start.y ] := start.y
      window[ w.size.y  ] := size.y + 1
      window[ w.start ] := (screen.width * start.y) + window[ w.start.x
]
      window[ w.size  ] := screen.width * window[ w.size.y ]
      window[ w.pixels.line ] := screen.width
      window[ w.foreground.color ] := 255
      window[ w.background.color ] := 0
      window[ w.cursor.x ] := 0
      window[ w.cursor.y ] := 0
  :
  --}}}
  --{{{  display.text
  PROC display.text ( [] INT window, [] BYTE screen,
                      VAL [] BYTE text, VAL [] INT font )

    s IS [screen FROM window[ w.start ] FOR window[ w.size ]] :
    SEQ
      window[ w.cursor.x ] := 0
      window[ w.cursor.y ] := 0
      write.string( window, s, text, font )
  :
  --}}}
  --{{{  place.numbers
  PROC place.numbers ( [][] BYTE screen, [][][] BYTE char.array,
                       VAL [] BYTE string,
                       VAL INT start.x, start.y, size.x, size.y )

    INT x :
    SEQ
      x := start.x
      SEQ i = 0 FOR SIZE string
        VAL char IS INT string[i] :
        SEQ
          IF
            --{{{  display space character
            char = (INT ' ')
              VAL source IS char.array[11] :
              MOVE2D( source, 0, 0, screen, x, start.y, size.x, size.y )
            --}}}
            --{{{  display decimal character
            char = (INT '.')
              VAL source IS char.array[10] :
              MOVE2D( source, 0, 0, screen, x, start.y, size.x, size.y )
            --}}}
            --{{{  display number character
            TRUE
              VAL source IS char.array[char - (INT '0')] :
              MOVE2D( source, 0, 0, screen, x, start.y, size.x, size.y )
            --}}}
          x := x + size.x
  :
  --}}}
  --{{{  MAX
  REAL32 FUNCTION MAX ( VAL REAL32 a, b )

    REAL32 r :
```

```
    VALOF
      IF
        a > b
          r := a
        TRUE
          r := b
      RESULT r
:
--}}}
--{{{  MIN
REAL32 FUNCTION MIN ( VAL REAL32 a, b )

  REAL32 r :
  VALOF
    IF
      a < b
        r := a
      TRUE
        r := b
    RESULT r
:
--}}}
--{{{  IMAX
INT FUNCTION IMAX ( VAL INT a, b )

  INT r :
  VALOF
    IF
      a > b
        r := a
      TRUE
        r := b
    RESULT r
:
--}}}
--{{{  IMIN
INT FUNCTION IMIN ( VAL INT a, b )

  INT r :
  VALOF
    IF
      a < b
        r := a
      TRUE
        r := b
    RESULT r
:
--}}}
--}}}
--{{{  variables
[w.length] INT window :
[4][w.length] INT text.window :
[12][char.height][char.width] BYTE number :

CHAN OF ANY synch, synch1 :
--}}}
VAL font IS SYS6 :
SEQ
  --{{{  initialize
  set.B408( 0, 0, 0, 0, 0 )
  --{{{  set up B409
  set.timing( toPrev, width, height, line.frequency,
              frame.rate, pixel.clock, interlace )
  spectrum( toPrev )
```

```
    --}}}
    --{{{  generate display table for numbers
    [w.length] INT window :
    [12][char.width*char.height] BYTE n RETYPES number :
    SEQ
      SEQ i = 0 FOR 12
        SEQ j = 0 FOR char.width*char.height
          n[i][j] := 0 (BYTE)
      window := [ 0, char.width*char.height, char.width,
                  0, 0, char.width, char.height, 255, 0, 0, 0 ]
      SEQ i = 0 FOR 10
        display.text( window, n[i], [ BYTE( i+(INT '0') )], font )
      display.text( window, n[10], ".", font )
      display.text( window, n[11], " ", font )
    --}}}
    --{{{  initial display
    [] INT int.screen RETYPES screen.map :
    SEQ i = 0 FOR (20*65536)/4
      int.screen[i] := 0

    window := [ 0, screen.size, screen.width, 0, 0,
                screen.width, screen.height, 255, 0, 0, 0 ]

    screen IS [screen.map FROM screen.offset[0] FOR screen.size ] :
    SEQ
      --{{{  draw first window
      draw.line( window, screen,  31, 111, 288, 111, 255(BYTE) )
      draw.line( window, screen, 288, 111, 288, 368, 255(BYTE) )
      draw.line( window, screen,  31, 368, 288, 368, 255(BYTE) )
      draw.line( window, screen,  31, 111,  31, 368, 255(BYTE) )
      --}}}
      --{{{  draw second window
      draw.line( window, screen, 351, 111, 608, 111, 255(BYTE) )
      draw.line( window, screen, 608, 111, 608, 368, 255(BYTE) )
      draw.line( window, screen, 351, 368, 608, 368, 255(BYTE) )
      draw.line( window, screen, 351, 111, 351, 368, 255(BYTE) )
      --}}}
      --{{{  draw text
      SEQ
        center.string( window, screen, 320, 1, "FPA Seeker Emulator",
font )
        center.string( window, screen, 160, 70, "Raw FPA Image", font )
        center.string( window, screen, 480, 70, "Processed Image", font
)

        place.string( window, screen, 0, 400, "Range: ", font )
        place.string( window, screen, 0, 440, "Sim Time: ", font )
        place.string( window, screen, 350, 400, "Frame Rate:", font )
        place.string( window, screen, 350, 440, "Frame Number:", font )

        set.text.window( text.window[0], "Range:", "0123456.789", font,
16,400, 32 )
        set.text.window( text.window[1], "Sim Time: ", "123.456", font,
16,440, 32 )
        set.text.window( text.window[2], "Frame Rate:    ", "123", font,
350,400, 32 )
        set.text.window( text.window[3], "Frame Number: ", "123", font,
350,440, 32 )

        --{{{  COMMENT place initial numbers
        --:::A  0 0
        --{{{  place initial numbers
        --[screen.height][screen.width] BYTE s2 RETYPES screen :
        --SEQ
```

```
            --place.numbers( s2,    number,    "                    0.000",
text.window[0][w.start.x],
            --                          text.window[0][w.start.y],  char.width,
char.height )
              --place.numbers( s2,      number,      "              0.000",
text.window[1][w.start.x],
            --                          text.window[1][w.start.y],  char.width,
char.height )
            --place.numbers( s2, number, "  0", text.window[2][w.start.x],
            --                          text.window[2][w.start.y],  char.width,
char.height )
            --place.numbers( s2, number, "  0", text.window[3][w.start.x],
            --                          text.window[3][w.start.y],  char.width,
char.height )
          --}}}
          --}}}
        --}}}
        [screen.map FROM screen.offset[1] FOR screen.size] := screen
        [screen.map FROM screen.offset[2] FOR screen.size] := screen
        [screen.map FROM screen.offset[3] FOR screen.size] := screen
      --}}}
      set.B408( 0, 0, 0, 1, 0 )
      --}}}
      PRI PAR
        --{{{  synchronize display to event
        PAR
          EventProc( Event, synch1  )
          Buffer( synch, synch1 )
        --}}}
        --{{{  run system
        --{{{  variables
        INT load :
        INT frame :
        REAL32 range, time, rate :
        INT len :
        [12] BYTE string, string1 :

        --{{{  command variables
        BYTE length :
        [max.message] INT message :
        command IS message[0] :
        params IS [message FROM 1 FOR max.message-1] :
        [] REAL32 r.params RETYPES params :
        --}}}
        --}}}
        SEQ
          load := 0
          WHILE TRUE
            SEQ
              fromSP ? length::message
              IF
                --{{{  read graphics display
                command = c.read.graphics
                  who IS params[0] :
                  CASE who
                    track.display
                      --{{{
                        screen IS   [screen.map   FROM   DisplayStart   FOR
screen.size ] :
                      [screen.height][screen.width] BYTE s2 RETYPES screen
:
                      SEQ
                        toSP ! screen.height
                        SEQ i = 0 FOR screen.height
```

```
                    toSP ! screen.width::s2[i]
                --}}}
              display.palette
                --{{{
                SKIP
                --}}}
              ELSE
                --{{{  image.display 0 or 1
                INT number.of.transfers, bufLength :
                [maxGraphicBuffer]BYTE graphicsBuffer :
                SEQ
                  toNext   ! length::message
                  fromNext ? number.of.transfers
                  toSP     ! number.of.transfers
                  SEQ i = 0 FOR number.of.transfers
                    SEQ
                      fromNext ? bufLength::graphicsBuffer
                      toSP     ! bufLength::graphicsBuffer
                --}}}
            --}}}
            --{{{  otherwise, normal processing
            TRUE
              SEQ
                --{{{  place range, time, frame number, and frame rate
on screen
                 screen IS  [screen.map  FROM  screen.offset[load]  FOR
screen.size ] :
                range   IS  r.params[1] :
                time    IS  r.params[2] :
                frame   IS  params[3]   :
                rate    IS  r.params[4] :
                SEQ
                  IF
                    frame >= 0
                      --{{{  display numbers
                       [screen.height][screen.width] BYTE  s2  RETYPES
screen :
                      SEQ
                        --{{{  range
                           range :=    MAX(    0.001(REAL32),    MIN(
9999999.999(REAL32), range ) )
                        REAL32TOSTRING( len, string, range, 7, 3 )
                        place.numbers( s2, number, [string FROM 1 FOR
11], text.window[0][w.start.x],
                                       text.window[0][w.start.y], 16,
32 )
                        --}}}
                        --{{{  time
                           time :=    MAX(    0.001(REAL32),    MIN(
999.999(REAL32), time ) )
                        REAL32TOSTRING( len, string, time, 3, 3 )
                        place.numbers( s2, number, [string FROM 1 FOR
7], text.window[1][w.start.x],
                                       text.window[1][w.start.y], 16,
32 )
                        --}}}
                        --{{{  frame rate
                        rate := MAX( 0.0(REAL32), MIN( 999.0(REAL32),
rate ) )
                        INTTOSTRING( len, string, INT ROUND rate )
                        [string FROM 0 FOR 3] := "   "
                        [string FROM 3-len FOR len] := [string FROM 0
FOR len]
```

```
                            place.numbers( s2, number, [string FROM 0 FOR
3], text.window[2][w.start.x],
                                                text.window[2][w.start.y], 16,
32 )
                            --}}}
                            --{{{  frame number
                            frame := IMAX( 0, IMIN( 999, frame ) )
                            INTTOSTRING( len, string1, frame )
                            [string FROM 0 FOR 3] := "   "
                            [string FROM 3-len FOR len] := [string1 FROM 0
FOR len]
                            place.numbers( s2, number, [string FROM 0 FOR
3], text.window[3][w.start.x],
                                                text.window[3][w.start.y], 16,
32 )
                            --}}}
                          --}}}
                        TRUE
                          --{{{  clear the text windows
                          SEQ i = 0 FOR 4
                            win IS text.window[i] :
                            clear.window( win, [screen FROM win[ w.start ]
FOR win[ w.size ]] )
                          --}}}
                    --}}}
                    --{{{  update screen
                    IF
                      params[0] = 1
                        synch ! screen.address[load]
                      TRUE
                        DisplayStart := screen.address[load]
                    --}}}
                    load := (load + 1) /\ 3
                --}}}

        --}}}
:
```

6.2.1.26. PROC XBar "xbar.occ"

```
PROC XBar (CHAN OF ANY fromGuidance, toGuidance)
  #INCLUDE "s_header.inc"
  --{{{  hardware values
  VAL XbarBase IS #0300 :

  PORT OF INT xbar.data :
  PLACE xbar.data AT XbarBase :

  PORT OF INT xbar.status :
  PLACE xbar.status AT XbarBase + 1 :

  CHAN OF INT Event :
  PLACE Event AT 8 :
  --}}}
  --{{{  constants
  VAL local.c.guidance.initialize IS 0 :
  VAL local.c.guidance.run        IS 1 :
  --}}}
  --{{{  procs
  --{{{  INT.to.xbar   (VAL INT value)
  PROC INT.to.xbar (VAL INT value)
    INT test.status :
    SEQ
      xbar.status ? test.status
      WHILE ((test.status /\ #0001) = 1)
        xbar.status ? test.status
      xbar.data ! value
  :
  --}}}
  --{{{  INT.from.xbar (INT value)
  PROC INT.from.xbar (INT value)
    INT signal :
    SEQ
      Event ? signal
      xbar.data ? value
  :
  --}}}

  --{{{  vector.to.xbar   (VAL []INT value)
  PROC vector.to.xbar (VAL []INT value)
    SEQ i = 0 FOR SIZE(value)
      INT.to.xbar (value[i])
  :
  --}}}
  --{{{  vector.from.xbar ([]INT value)
  PROC vector.from.xbar ([]INT value)
    SEQ i = 0 FOR SIZE(value)
      INT.from.xbar (value[i])
  :
  --}}}
  --}}}

  SEQ
    --{{{  clear input buffer at start up
    INT dummy :
    xbar.data ? dummy
    --}}}
    WHILE TRUE
      INT32 command :
      SEQ
        fromGuidance ? command
```

```
        CASE (INT command)
          --{{{  initialize
          c.guidance.initialize
            INT dummy :
            SEQ
              INT.to.xbar (local.c.guidance.initialize)
              --{{{  send and receive dummy values to satisfy crossbar
sequencer
              vector.to.xbar ([0, 0, 0, 0, 0, 0, 0, 0, 0 ])
              vector.to.xbar ([0, 0, 0, 0, 0, 0, 0, 0, 0 ])
              [2]INT dummy :
              vector.from.xbar(dummy)
              --}}}
          --}}}
          --{{{  run
          c.guidance.run
            [18]INT data.vector : --REAL32 range, time INT16 x, y,
                                  --[3]REAL32 seeker.xyz, target.xyz
            [2]INT shift.values :
            SEQ
              fromGuidance ? data.vector
              INT.to.xbar (local.c.guidance.run)
              vector.to.xbar (data.vector)
              vector.from.xbar (shift.values)
              toGuidance ! shift.values

          --}}}
:
```

6.2.2. Include files

```
--{{{  system constants
VAL bpw       IS 4 :
VAL bpw.shift IS 2 :
VAL mint      IS MOSTNEG INT :

VAL screen.int.address      IS (#80100000   >< mint) >> bpw.shift :
VAL DisplayStart.address    IS (#00000000   >< mint) >> bpw.shift :
VAL InterlaceEnable.address IS (#000C0000   >< mint) >> bpw.shift :
VAL EventMode.address       IS (#00100000   >< mint) >> bpw.shift :
VAL OutputEnable.address    IS (#00140000   >< mint) >> bpw.shift :
VAL Ready.address           IS (#00040000   >< mint) >> bpw.shift :
--}}}
--{{{  window constants
VAL w.start             IS   0 :
VAL w.size              IS   1 :
VAL w.pixels.line       IS   2 :
VAL w.start.x           IS   3 :
VAL w.start.y           IS   4 :
VAL w.size.x            IS   5 :
VAL w.size.y            IS   6 :
VAL w.foreground.color  IS   7 :
VAL w.background.color  IS   8 :
VAL w.cursor.x          IS   9 :
VAL w.cursor.y          IS  10 :
VAL w.length            IS  11 :
--}}}
--{{{  text drawing modes
VAL normal.mode       IS 0 :
VAL foreground.mode   IS 4 :
VAL and.mode          IS 1 :
VAL or.mode           IS 2 :
VAL xor.mode          IS 3 :
--}}}
--{{{  window decisions
VAL in.range        IS 0 :
VAL part.inrange    IS 1 :
VAL not.inrange     IS 2 :
--}}}
--{{{  font file format
VAL dfVersion.p         IS 0 :    --2
VAL dfSize.p            IS 2 :    --4
VAL dfCopyright.p       IS 6 :    --60
VAL dfType.p            IS 66 :   --2
VAL dfPoints.p          IS 68 :   --2
VAL dfVertRes.p         IS 70 :   --2
VAL dfHorizRes.p        IS 72 :   --2
VAL dfAscent.p          IS 74 :   --2
VAL dfInternalLeading.p IS 76 :   --2
VAL dfExternalLeading.p IS 78 :   --2
VAL dfItalic.p          IS 80 :   --1
VAL dfUnderline.p       IS 81 :   --1
VAL dfStrikeOut.p       IS 82 :   --1
VAL dfWeight.p          IS 83 :   --2
VAL dfCharSet.p         IS 85 :   --1
VAL dfPixWidth.p        IS 86 :   --2
VAL dfPixHeight.p       IS 88 :   --2
VAL dfPitchAndFamily.p  IS 90 :   --1
VAL dfAvgWidth.p        IS 91 :   --2
VAL dfMaxWidth.p        IS 93 :   --2
VAL dfFirstChar.p       IS 95 :   --1
VAL dfLastChar.p        IS 96 :   --1
```

```
VAL dfDefaultChar.p      IS 97 :    --1
VAL dfBreakChar.p        IS 98 :    --1
VAL dfWidthBytes.p       IS 99 :    --2
VAL dfDevice.p           IS 101 :   --4
VAL dfFace.p             IS 105 :   --4
VAL dfBitsPointer.p      IS 109 :   --4
VAL dfBitsOffset.p       IS 113 :   --4
VAL CharTable.p          IS 118 :   --
--}}}
--{{{  font specification offsets
VAL  fs.PixWidth     IS  0 :
VAL  fs.PixHeight    IS  1 :
VAL  fs.FirstChar    IS  2 :
VAL  fs.LastChar     IS  3 :
VAL  fs.BitsOffset   IS  4 :
VAL  fs.size         IS  5 :
--}}}
```

```
--{{{  misc. constants
VAL  max.message      IS  205 :
VAL  max.frames       IS  200 :
VAL  max.sim.frames   IS 2000 :
--}}}
--{{{  commands
--  Controller commands                          -- parameters for commands
VAL  c.start.frame              IS    0 :      -- frame
VAL  c.run.single               IS    1 :
VAL  c.run.continuous           IS    2 :
VAL  c.frame.rate                 IS    3 :    -- start; frames; [frames]
data
VAL  c.frame.time                 IS    4 :    -- start; frames; [frames]
data
VAL  c.frame.range                IS    5 :    -- start; frames; [frames]
data
VAL  c.sim.position             IS    6 :      -- variable; start; frames; []
data
VAL  c.sim.start.frames         IS    7 :      -- first.frame; last.frame
VAL  c.test.controller          IS    8 :
VAL  c.restart                  IS    9 :
VAL  c.set.calibration          IS   10 :      -- back.level0; back.level1;
                                               -- sp.level0; sp.level1


--  GTSEI commands
VAL  c.set.crossbar             IS  256 :      -- shift

--  Target commands
VAL     c.set.target                                    IS     512   :     --
frame;subpixel;row.shift;col.shift
VAL     c.target.row                                    IS     513   :     --
frame;subpixel;row;[128]data
VAL  c.test.target              IS  514 :

--  Background commands
VAL  c.set.background           IS  768 :      -- frame
VAL  c.background.row           IS  769 :      -- frame; row; [128] data
VAL  c.gain.row                 IS  770 :      -- row; [128] data
VAL  c.offset.row               IS  771 :      -- row; [128] data
VAL  c.global.scale             IS  772 :      -- scale
VAL  c.test.background          IS  773 :
VAL  c.calibration.frame        IS  774 :      -- calibration.level

--  Signal Processing commands
VAL   c.sp.frame                        IS 1024 :   -- calibration.frame;
calibration.level;
                                                        -- lower.threshold;
upper.threshold;
                                                   -- frame.range; frame.time;
                                                   -- frame.number; frame.rate;
                                                   -- [6]seeker.target.position

--  Guidance commands
VAL  c.guidance.initialize      IS 1280 :
VAL  c.guidance.set.mode        IS 1281 :      -- mode
VAL   c.guidance.run                    IS 1282 :   -- FPA; frame.range;
frame.time;
                                                        -- column.hot.spot;
row.hot.spot;
                                                   -- [3]seeker.position;
                                                   -- [3]target.position


--  Track Display commands
VAL   c.display.info                    IS 1536 :   -- FPA; frame.range;
frame.time;
```

```
                                                    -- frame.number; frame.rate

-- command for reading graphics buffers or palette
VAL  c.read.graphics           IS 2000 :  -- display
--}}}
--{{{  commands for continuous frame operation
VAL  cc.exit          IS  0 :
VAL  cc.shift.image   IS  1 :  -- column.shift; row.shift
--}}}
--{{{  variables in position
VAL  p.seeker.x     IS  0 :
VAL  p.seeker.y     IS  1 :
VAL  p.seeker.z     IS  2 :
VAL  p.target.x     IS  3 :
VAL  p.target.y     IS  4 :
VAL  p.target.z     IS  5 :
VAL  p.length       IS  6 :
--}}}
--{{{  guidance mode constants
VAL  gm.none       IS   0 :
VAL  gm.external   IS   1 :
VAL  gm.internal   IS   2 :
--}}}
--{{{  graphics read constants
VAL image.display.0 IS 0 :
VAL image.display.1 IS 1 :
VAL track.display   IS 2 :
VAL display.palette IS 3 :

VAL maxGraphicBuffer IS 640 :
--}}}
```

```
VAL SYS6 IS [#44DA0200,#6F430000,#69727970,#20746867,
#20296328,#37383931,#6F535A20,#43207466,#6F70726F,#69746172,#6E6F    ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#12C0008 ,
#18012C  ,#2       ,#90000000,#10FF01  ,#10300020,#1001000 ,#1F2DFF  ,
#47602   ,#4A800   ,#0       ,#4DA00   ,#100000  ,#1004DA  ,#10051A  ,
#10055A  ,#10059A  ,#1005DA  ,#10061A  ,#10065A  ,#10069A  ,#1006DA  ,
#10071A  ,#10075A  ,#10079A  ,#1007DA  ,#10081A  ,#10085A  ,#10089A  ,
#1008DA  ,#10091A  ,#10095A  ,#10099A  ,#1009DA  ,#100A1A  ,#100A5A  ,
#100A9A  ,#100ADA  ,#100B1A  ,#100B5A  ,#100B9A  ,#100BDA  ,#100C1A  ,
#100C5A  ,#100C9A  ,#100CDA  ,#100D1A  ,#100D5A  ,#100D9A  ,#100DDA  ,
#100E1A  ,#100E5A  ,#100E9A  ,#100EDA  ,#100F1A  ,#100F5A  ,#100F9A  ,
#100FDA  ,#10101A  ,#10105A  ,#10109A  ,#1010DA  ,#10111A  ,#10115A  ,
#10119A  ,#1011DA  ,#10121A  ,#10125A  ,#10129A  ,#1012DA  ,#10131A  ,
#10135A  ,#10139A  ,#1013DA  ,#10141A  ,#10145A  ,#10149A  ,#1014DA  ,
#10151A  ,#10155A  ,#10159A  ,#1015DA  ,#10161A  ,#10165A  ,#10169A  ,
#1016DA  ,#10171A  ,#10175A  ,#10179A  ,#1017DA  ,#10181A  ,#10185A  ,
#10189A  ,#1018DA  ,#10191A  ,#10195A  ,#10199A  ,#1019DA  ,#101A1A  ,
#101A5A  ,#101A9A  ,#101ADA  ,#101B1A  ,#101B5A  ,#101B9A  ,#101BDA  ,
#101C1A  ,#101C5A  ,#101C9A  ,#101CDA  ,#101D1A  ,#101D5A  ,#101D9A  ,
#101DDA  ,#101E1A  ,#101E5A  ,#101E9A  ,#101EDA  ,#101F1A  ,#101F5A  ,
#101F9A  ,#101FDA  ,#10201A  ,#10205A  ,#10209A  ,#1020DA  ,#10211A  ,
#10215A  ,#10219A  ,#1021DA  ,#10221A  ,#10225A  ,#10229A  ,#1022DA  ,
#10231A  ,#10235A  ,#10239A  ,#1023DA  ,#10241A  ,#10245A  ,#10249A  ,
#1024DA  ,#10251A  ,#10255A  ,#10259A  ,#1025DA  ,#10261A  ,#10265A  ,
#10269A  ,#1026DA  ,#10271A  ,#10275A  ,#10279A  ,#1027DA  ,#10281A  ,
#10285A  ,#10289A  ,#1028DA  ,#10291A  ,#10295A  ,#10299A  ,#1029DA  ,
#102A1A  ,#102A5A  ,#102A9A  ,#102ADA  ,#102B1A  ,#102B5A  ,#102B9A  ,
#102BDA  ,#102C1A  ,#102C5A  ,#102C9A  ,#102CDA  ,#102D1A  ,#102D5A  ,
#102D9A  ,#102DDA  ,#102E1A  ,#102E5A  ,#102E9A  ,#102EDA  ,#102F1A  ,
#102F5A  ,#102F9A  ,#102FDA  ,#10301A  ,#10305A  ,#10309A  ,#1030DA  ,
#10311A  ,#10315A  ,#10319A  ,#1031DA  ,#10321A  ,#10325A  ,#10329A  ,
#1032DA  ,#10331A  ,#10335A  ,#10339A  ,#1033DA  ,#10341A  ,#10345A  ,
#10349A  ,#1034DA  ,#10351A  ,#10355A  ,#10359A  ,#1035DA  ,#10361A  ,
#10365A  ,#10369A  ,#1036DA  ,#10371A  ,#10375A  ,#10379A  ,#1037DA  ,
#10381A  ,#10385A  ,#10389A  ,#1038DA  ,#10391A  ,#10395A  ,#10399A  ,
#1039DA  ,#103A1A  ,#103A5A  ,#103A9A  ,#103ADA  ,#103B1A  ,#103B5A  ,
#103B9A  ,#103BDA  ,#103C1A  ,#103C5A  ,#103C9A  ,#103CDA  ,#103D1A  ,
#103D5A  ,#103D9A  ,#103DDA  ,#103E1A  ,#103E5A  ,#103E9A  ,#103EDA  ,
#103F1A  ,#103F5A  ,#103F9A  ,#103FDA  ,#10401A  ,#10405A  ,#10409A  ,
#1040DA  ,#10411A  ,#10415A  ,#10419A  ,#1041DA  ,#10421A  ,#10425A  ,
#10429A  ,#1042DA  ,#10431A  ,#10435A  ,#10439A  ,#1043DA  ,#10441A  ,
#10445A  ,#449A    ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#74737953,#36206D65,#34206400,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#1C1E0F07,
#76763838,#70707676,#77777070,#1E1C3B3B,#70F     ,#0       ,#0       ,
#3878F0E0,#6E6E1C1C,#E0E6E6E ,#EEEE0E0E,#7838DCDC,#E0F0    ,#0       ,
#0       ,#1F1F0F07,#79793F3F,#7F7F7979,#78787F7F,#1F1F3C3C,#70F     ,
#0       ,#0       ,#F8F8F0E0,#9E9EFCFC,#FEFE9E9E,#1E1EFEFE,#F8F83C3C,
#E0F0    ,#0       ,#0       ,#7F7F3E1C,#7F7F7F7F,#1F1F3F3F,#7070F0F ,
#1010303 ,#0       ,#0       ,#0       ,#7F7F3E1C,#FFFFFFFF,#FCFCFEFE,
#F0F0F8F8,#C0C0E0E0,#8080    ,#0       ,#0       ,#1010000 ,#7070303 ,
#1F1F0F0F,#7070F0F ,#1010303 ,#0       ,#0       ,#0       ,#C0C08080,
#F0F0E0E0,#FCFCF8F8,#F0F0F8F8,#C0C0E0E0,#8080    ,#0       ,#0       ,
#7070301 ,#3030707 ,#7F7F7D39,#397D7F7F,#1010101 ,#303     ,#0       ,
#0       ,#F0F0E0C0,#E0E0F0F0,#FFFFDFCE,#CEDFFFFF,#C0C0C0C0,#E0E0    ,
#0       ,#0       ,#1010000 ,#7070303 ,#1F1F0F0F,#1D3D3F3F,#1010101 ,
#303     ,#0       ,#0       ,#C0C08080,#F0F0E0E0,#FCFCF8F8,#DCDEFEFE,
#C0C0C0C0,#E0E0    ,#0       ,#0       ,#0       ,#0       ,#7070703 ,
#3070707 ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#E0E0E0C0,#C0E0E0E0,#0       ,#0       ,#0       ,#0       ,#FFFFFFFF,
#FFFFFFFF,#F8F8F8FC,#FCF3F6F8,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFF    ,
#FFFFFFFF,#FFFFFFFF,#1F1F1F3F,#3F1F1F1F,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,
#FFFF    ,#0       ,#0       ,#1C1E0F07,#38383838,#70F1E1C ,#0       ,
```

```
#0       ,#0       ,#0       ,#0       ,#3878F0E0,#1C1C1C1C,#E0F07838,
#0       ,#0       ,#0       ,#FFFFFFFF,#FFFFFFFF,#E3E1F0F8,#C7C7C7C7,
#F8F0E1E3,#FFFFFFFF,#FFFFFFFF,#FFFF    ,#FFFFFFFF,#FFFFFFFF,#C7870F1F,
#E3E3E3E3,#1F0F87C7,#FFFFFFFF,#FFFFFFFF,#FFFF    ,#0       ,#303     ,
#1F0F0000,#70703838,#38387070,#F1F     ,#0       ,#0       ,#0       ,
#3E1EFEFE,#E6E67676,#7070E0E0,#E0E07070,#80C0    ,#0       ,#0       ,
#E0E0703 ,#1C1C1C1C,#3070E0E ,#1010101 ,#1010707 ,#101     ,#0       ,
#0       ,#3838F0E0,#1C1C1C1C,#E0F03838,#C0C0C0C0,#C0C0F0F0,#C0C0    ,
#0       ,#0       ,#3030303 ,#3030303 ,#3030303 ,#1F0F0303,#F1F3F3F ,
#0       ,#0       ,#0       ,#9C9CF8F0,#80808E8E,#80808080,#80808080,
#808080  ,#0       ,#0       ,#0       ,#7070707 ,#7070707 ,#7070707 ,
#3F1F0707,#1E3F7F7F,#0       ,#0       ,#0       ,#707FFFF ,#707FFFF ,
#7070707 ,#7070707 ,#7F7F3F1F,#1E3F    ,#0       ,#0       ,#39390101,
#70F1D1D ,#7C7C1C0E,#F070E1C ,#39391D1D,#101     ,#0       ,#0       ,
#CECEC0C0,#F0F8DCDC,#1F1F1C38,#F8F0381C,#CECEDCDC,#C0C0    ,#0       ,
#0       ,#3E3C3830,#3F3F3F3F,#3F3F3F3F,#3F3F3F3F,#3C3E3F3F,#3038    ,
#0       ,#0       ,#0       ,#E0C08000,#FCFCF8F0,#C0E0F0F8,#80      ,
#0       ,#0       ,#0       ,#0       ,#7030100 ,#3F3F1F0F,#3070F1F ,
#1       ,#0       ,#0       ,#0       ,#7C3C1C0C,#FCFCFCFC,#FCFCFCFC,
#FCFCFCFC,#3C7CFCFC,#C1C     ,#0       ,#0       ,#1010000 ,#7070303 ,
#1010F0F ,#F0F0101 ,#3030707 ,#101     ,#0       ,#0       ,#C0C08080,
#F0F0E0E0,#C0C0F8F8,#F8F8C0C0,#E0E0F0F0,#8080C0C0,#0       ,#0       ,
#3C3C3C18,#3C3C3C3C,#183C3C3C,#18181818,#3C180000,#183C    ,#0       ,
#0       ,#3C3C3C18,#3C3C3C3C,#183C3C3C,#18181818,#3C180000,#183C    ,
#0       ,#0       ,#737B3F1F,#73737373,#7B737373,#3031F3F ,#3030303 ,
#303     ,#0       ,#0       ,#9C9CFFFF,#9C9C9C9C,#9C9C9C9C,#9C9C9C9C,
#9C9C9C9C,#9C9C    ,#0       ,#0       ,#383C1F0F,#E1C3838 ,#1C1C0F07,
#70F     ,#1E1C0000,#70F     ,#0       ,#0       ,#3878F0E0,#0       ,
#3838F0E0,#3870E0F0,#3C1C1C1C,#F0F8    ,#0       ,#0       ,#0       ,
#0       ,#0       ,#F0F0F0F ,#F0F     ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#F0F0F0F0,#F0F0    ,#0       ,#0       ,
#0       ,#1010000 ,#7070303 ,#1010F0F ,#F0F0101 ,#3030707 ,#1F000101,
#0       ,#0       ,#C0C08080,#F0F0E0E0,#C0C0F8F8,#F8F8C0C0,#E0E0F0F0,
#FC80C0C0,#0       ,#0       ,#1010000 ,#7070303 ,#1010F0F ,#1010101 ,
#1010101 ,#101     ,#0       ,#0       ,#C0C08080,#F0F0E0E0,#C0C0F8F8,
#C0C0C0C0,#C0C0C0C0,#C0C0    ,#0       ,#0       ,#1010000 ,#1010101 ,
#1010101 ,#F0F0101 ,#3030707 ,#101     ,#0       ,#0       ,#C0C00000,
#C0C0C0C0,#C0C0C0C0,#F8F8C0C0,#E0E0F0F0,#8080C0C0,#0       ,#0       ,
#0       ,#1010101 ,#7F7F0101,#1010101 ,#101     ,#0       ,#0       ,
#0       ,#0       ,#F0E0C080,#FEFEFCF8,#E0F0F8FC,#80C0    ,#0       ,
#0       ,#0       ,#0       ,#F070301 ,#7F7F3F1F,#70F1F3F ,#103     ,
#0       ,#0       ,#0       ,#0       ,#80808C80,#FEFE8080,#80808080,
#8080    ,#0       ,#0       ,#0       ,#0       ,#38380000,#38383838,
#3F3F    ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#FCFC    ,#0       ,#0       ,#0       ,#0       ,#0       ,
#E060000 ,#7F7F3E1E,#60E1E3E ,#0       ,#0       ,#0       ,#0       ,
#0       ,#70600000,#FEFE7C78,#6070787C,#0       ,#0       ,#0       ,
#0       ,#0       ,#1010000 ,#7070303 ,#1F1F0F0F,#3F3F    ,#0       ,
#0       ,#0       ,#0       ,#80800000,#E0E0C0C0,#F8F8F0F0,#FCFC    ,
#0       ,#0       ,#0       ,#0       ,#3F3F0000,#F0F1F1F ,#3030707 ,
#101     ,#0       ,#0       ,#0       ,#0       ,#FCFC0000,#F0F0F8F8,
#C0C0E0E0,#8080    ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#F0F0F07 ,#F0F0F0F ,#70F0F0F ,#7070707 ,#F070000 ,#70F     ,#0       ,
#0       ,#80808000,#80808080,#808080  ,#0       ,#80000000,#80      ,
#0       ,#0       ,#E0E0E0E ,#60606   ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#70707070,#606060  ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#E0E0000 ,#3F3F0E0E,#E0E0E0E ,
#3F3F0E0E,#E0E0E0E ,#0       ,#0       ,#0       ,#70700000,#FCFC7070,
#70707070,#FCFC7070,#70707070,#0       ,#0       ,#0       ,#1010101 ,
#393D1F0F,#F1F3D39 ,#1010101 ,#1011F1F ,#101     ,#0       ,#0       ,
#C0C0C0C0,#C0C0FCFC,#FCF9C0C0,#DECECEDE,#C0C0F8FC,#C0C0    ,#0       ,
#0       ,#33333F1E,#1E3F    ,#1010000 ,#7070303 ,#1C1C0E0E,#3838    ,
```

```
#0       ,#0       ,#1C1C0E0E,#70703838,#C0C0E0E0,#8080    ,#66667E3C,
#3C7E    ,#0       ,#0       ,#1C1C0F07,#1C1C1C1C,#70F1D1D ,#713B1F0F,
#38707070,#F1F     ,#0       ,#0       ,#E0E0C080,#E0E0E0E0,#8080C0C0,
#F89C0E00,#F870F0F0,#8EDE    ,#0       ,#0       ,#1030301 ,#3030100 ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#E0E0E0C0,
#80C0E0  ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#7070301 ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,#103     ,#0       ,
#0       ,#80F0F0  ,#0       ,#0       ,#0       ,#80000000,#F0F0    ,
#0       ,#0       ,#707     ,#0       ,#0       ,#0       ,#0       ,
#707     ,#0       ,#0       ,#70F0E0C0,#70707070,#70707070,#70707070,
#F0707070,#C0E0    ,#0       ,#0       ,#0       ,#E0E1C1C ,#7F7F0707,
#E0E0707 ,#1C1C    ,#0       ,#0       ,#0       ,#0       ,#70703838,
#FEFEE0E0,#7070E0E0,#3838    ,#0       ,#0       ,#0       ,#0       ,
#3030000 ,#3F3F0303,#3030303 ,#0       ,#0       ,#0       ,#0       ,
#0       ,#80800000,#F8F88080,#80808080,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#F070000 ,#703070F ,
#C0E     ,#0       ,#0       ,#0       ,#0       ,#0       ,#80000000,
#808080  ,#0       ,#0       ,#0       ,#0       ,#3F3F0000,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#F8F80000,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#F070000 ,#70F     ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#80000000,#80      ,#0       ,#0       ,
#0       ,#0       ,#1010000 ,#7070303 ,#1C1C0E0E,#3838    ,#0       ,
#0       ,#1C1C0E0E,#70703838,#C0C0E0E0,#8080    ,#0       ,#0       ,
#0       ,#0       ,#383C1F0F,#38383838,#39393838,#3E3E3B3B,#3C383C3C,
#F1F     ,#0       ,#0       ,#1C3CF8F0,#7C7C3C3C,#9C9CDCDC,#1C1C1C1C,
#3C1C1C1C,#F0F8    ,#0       ,#0       ,#F0F0703 ,#1010101 ,#1010101 ,
#1010101 ,#1010101 ,#1F1F    ,#0       ,#0       ,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#FCFC    ,#0       ,#0       ,#1C1E0F07,
#0       ,#0       ,#7030100 ,#38381C0E,#3F3F    ,#0       ,#0       ,
#1C3CF8F0,#1C1C1C1C,#70381C1C,#80C0E0  ,#0       ,#FCFC    ,#0       ,
#0       ,#1C1E0F07,#0       ,#3030000 ,#0       ,#1E1C0000,#70F     ,
#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,#F0F0381C,#1C1C1C38,#3C1C1C1C,
#F0F8    ,#0       ,#0       ,#3030100 ,#E0E0707 ,#38381C1C,#7F7F7070,
#0       ,#0       ,#0       ,#0       ,#F0F0F0F0,#70707070,#70707070,
#FEFE7070,#70707070,#7070    ,#0       ,#0       ,#38383F3F,#38383838,
#3F3F    ,#0       ,#3C380000,#F1F     ,#0       ,#0       ,#F8F8    ,
#0       ,#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,#0       ,
#383C1F0F,#38383838,#3F3F3838,#38383838,#3C383838,#F1F     ,#0       ,
#0       ,#3878F0E0,#0       ,#F8F00000,#1C1C1C3C,#3C1C1C1C,#F0F8    ,
#0       ,#0       ,#383F3F  ,#0       ,#0       ,#0       ,#3030101 ,
#303     ,#0       ,#0       ,#1C1CFCFC,#1C1C1C1C,#38381C1C,#E0E07070,
#8080C0C0,#8080    ,#0       ,#0       ,#383C1F0F,#38383838,#F0F1C38 ,
#3838381C,#3C383838,#F1F     ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,
#F0F0381C,#1C1C1C38,#3C1C1C1C,#F0F8    ,#0       ,#0       ,#383C1F0F,
#38383838,#F1F3C38 ,#0       ,#1C1C0000,#70F     ,#0       ,#0       ,
#1C3CF8F0,#1C1C1C1C,#FCFC1C1C,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,
#0       ,#0       ,#0       ,#70F0F07 ,#0       ,#F070000 ,#70F     ,
#0       ,#0       ,#0       ,#0       ,#808000  ,#0       ,#80000000,
#80      ,#0       ,#0       ,#0       ,#0       ,#70F0F07 ,#0       ,
#F070000 ,#703070F ,#C0E     ,#0       ,#0       ,#0       ,#808000  ,
#0       ,#80000000,#808080  ,#0       ,#0       ,#0       ,#1010000 ,
#7070303 ,#7070E0E ,#1010303 ,#0       ,#0       ,#0       ,#0       ,
#C0C0E0E0,#8080    ,#0       ,#C0C08080,#E0E0    ,#0       ,#0       ,
#0       ,#0       ,#3F3F    ,#3F3F0000,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#FCFC    ,#FCFC0000,#0       ,#0       ,
#0       ,#0       ,#0       ,#3030707 ,#101     ,#0       ,#3030101 ,
#707     ,#0       ,#0       ,#0       ,#80800000,#E0E0C0C0,#E0E07070,
#8080C0C0,#0       ,#0       ,#0       ,#1C1E0F07,#0       ,#1000000 ,
#3030303 ,#7030000 ,#307     ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,
#C0E07038,#80808080,#C0800000,#80C0    ,#0       ,#0       ,#1C1E0F07,
#39383838,#3B3B3B3B,#38393B3B,#1E1C3838,#70F     ,#0       ,#0       ,
#3878F0E0,#FCFC1C1C,#9C9C9C9C,#FCFC9C9C,#0       ,#F0F0    ,#0       ,
#0       ,#E0E0703 ,#1C1C1C1C,#38383838,#38383F3F,#38383838,#3838    ,
```

```
#0        ,#0        ,#7070E0C0,#38383838,#1C1C1C1C,#1C1CFCFC,#1C1C1C1C,
#1C1C     ,#0        ,#0        ,#38383F3F,#38383838,#3F3F3838,#38383838,
#38383838,#3F3F     ,#0        ,#0        ,#3870E0C0,#38383838,#E0C0E070,
#1C1C3870,#381C1C1C,#E0F0     ,#0        ,#0        ,#383C1F0F,#38383838,
#38383838,#38383838,#3C383838,#F1F      ,#0        ,#0        ,#1C3CF8F0,
#0        ,#0        ,#0        ,#3C1C0000,#F0F8     ,#0        ,#0        ,
#38383F3F,#38383838,#38383838,#38383838,#38383838,#3F3F     ,#0        ,
#0        ,#1C3CF8F0,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#3C1C1C1C,#F0F8     ,
#0        ,#0        ,#1C1C1F1F,#1C1C1C1C,#1F1F1C1C,#1C1C1C1C,#1C1C1C1C,
#1F1F     ,#0        ,#0        ,#FCFC     ,#0        ,#F0F00000,#0        ,
#0        ,#FCFC     ,#0        ,#0        ,#1C1C1F1F,#1C1C1C1C,#1F1F1C1C,
#1C1C1C1C,#1C1C1C1C,#1C1C     ,#0        ,#0        ,#FCFC     ,#0        ,
#F0F00000,#0        ,#0        ,#0        ,#0        ,#0        ,#383C1F0F,
#38383838,#38383838,#38383838,#3C383838,#F1F      ,#0        ,#0        ,
#3878F0E0,#0        ,#FCFC0000,#1C1C1C1C,#1C1C1C1C,#FCFC     ,#0        ,
#0        ,#38383838,#38383838,#3F3F3838,#38383838,#38383838,#3838     ,
#0        ,#0        ,#1C1C1C1C,#1C1C1C1C,#FCFC1C1C,#1C1C1C1C,#1C1C1C1C,
#1C1C     ,#0        ,#0        ,#1010F0F0 ,#1010101 ,#1010101 ,#1010101 ,
#1010101 ,#F0F      ,#0        ,#0        ,#C0C0F8F8,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#F8F8     ,#0        ,#0        ,#0        ,#0        ,
#0        ,#0        ,#1E1C1C00,#70F      ,#0        ,#0        ,#1C1C1C1C,
#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#3C1C1C1C,#F0F8     ,#0        ,#0        ,
#38383838,#39393838,#3F3F3B3B,#39393B3B,#38383838,#3838     ,#0        ,
#0        ,#70703838,#C0C0E0E0,#8080     ,#C0C08080,#7070E0E0,#3838     ,
#0        ,#0        ,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,
#1F1F     ,#0        ,#0        ,#0        ,#0        ,#0        ,#0        ,
#0        ,#FCFC     ,#0        ,#0        ,#70706060,#7C7C7878,#77777E7E,
#71717373,#70707070,#7070     ,#0        ,#0        ,#7070303 ,#1F1F0F0F,
#77773F3F,#C7C7E7E7,#7078787 ,#707      ,#0        ,#0        ,#3C3C3838,
#3F3F3E3E,#39393B3B,#38383838,#38383838,#3838     ,#0        ,#0        ,
#1C1C1C1C,#1C1C1C1C,#DCDC9C9C,#7C7CFCFC,#1C1C3C3C,#1C1C     ,#0        ,
#0        ,#383C1F0F,#38383838,#38383838,#38383838,#3C383838,#F1F      ,
#0        ,#0        ,#1C3CF8F0,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#3C1C1C1C,
#F0F8     ,#0        ,#0        ,#38383F3F,#38383838,#3F3F3838,#38383838,
#38383838,#3838     ,#0        ,#0        ,#1C3CF8F0,#1C1C1C1C,#F0F83C1C,
#0        ,#0        ,#0        ,#0        ,#0        ,#1C1E0F07,#38383838,
#38383838,#39383838,#1E1C3839,#70F      ,#0        ,#0        ,#3878F0E0,
#1C1C1C1C,#1C1C1C1C,#DC1C1C1C,#F878FCDC,#9CDC     ,#0        ,#0        ,
#38383F3F,#38383838,#3F3F3838,#38383939,#38383838,#3838     ,#0        ,
#0        ,#1C3CF8F0,#1C1C1C1C,#F0F83C1C,#E0E0C0C0,#38387070,#1C1C     ,
#0        ,#0        ,#381C0F07,#1C383838,#103070E ,#0        ,#3C380000,
#F1F      ,#0        ,#0        ,#1C3CF8F0,#0        ,#C0800000,#1C3870E0,
#381C1C1C,#E0F0     ,#0        ,#0        ,#3033F3F ,#3030303 ,#3030303 ,
#3030303 ,#3030303 ,#303      ,#0        ,#0        ,#8080F8F8,#80808080,
#80808080,#80808080,#80808080,#8080     ,#0        ,#0        ,#38383838,
#38383838,#38383838,#38383838,#1C383838,#70F      ,#0        ,#0        ,
#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#381C1C1C,#E0F0     ,#0        ,
#0        ,#38383838,#38383838,#38383838,#E0E1C1C ,#3030707 ,#101      ,
#0        ,#0        ,#E0E0E0E ,#E0E0E0E ,#E0E0E0E ,#38381C1C,#E0E07070,
#C0C0     ,#0        ,#0        ,#70707070,#70707070,#39393939,#3F3F3B3B,
#1E1E1F1F,#1C1C     ,#0        ,#0        ,#7070707 ,#87870707,#CECECECE,
#FEFEEEEE,#3C3C7C7C,#1C1C     ,#0        ,#0        ,#1C1C3838,#7070E0E ,
#1010303 ,#7070303 ,#1C1C0E0E,#3838     ,#0        ,#0        ,#1C1C0E0E,
#70703838,#C0C0E0E0,#7070E0E0,#1C1C3838,#E0E      ,#0        ,#0        ,
#38387070,#E0E1C1C ,#3030707 ,#1010101 ,#1010101 ,#101      ,#0        ,
#0        ,#E0E0707 ,#38381C1C,#E0E07070,#C0C0C0C0,#C0C0C0C0,#C0C0     ,
#0        ,#0        ,#3F3F     ,#0        ,#1010000 ,#7070303 ,#1C1C0E0E,
#3F3F     ,#0        ,#0        ,#1C1CFEFE,#70703838,#C0C0E0E0,#8080     ,
#0        ,#FEFE     ,#0        ,#0        ,#7070707 ,#7070707 ,#7070707 ,
#7070707 ,#7070707 ,#707      ,#0        ,#0        ,#F0F0     ,#0        ,
#0        ,#0        ,#0        ,#F0F0     ,#0        ,#0        ,#38387070,
#E0E1C1C ,#3030707 ,#101      ,#0        ,#0        ,#0        ,#0        ,
#0        ,#0        ,#80800000,#E0E0C0C0,#38387070,#1C1C     ,#0        ,
#0        ,#707      ,#0        ,#0        ,#0        ,#0        ,#707      ,
```

```
#0       ,#0       ,#7070F0F0,#70707070,#70707070,#70707070,#70707070,
#F0F0    ,#0       ,#0       ,#3030101 ,#E0E0707 ,#1C1C    ,#0       ,
#0       ,#0       ,#0       ,#0       ,#E0E0C0C0,#38387070,#1C1C    ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#7F7F    ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#FEFE    ,
#3010000 ,#1030303 ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#80C0E060,#C0E0E0C0,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#1C0F07  ,#1F0F0000,#3C38383C,
#F1F     ,#0       ,#0       ,#0       ,#0       ,#1C3CF8F0,#FCFC1C1C,
#1C1C1C1C,#FCFC    ,#0       ,#0       ,#38383838,#38383838,#38383F3F,
#38383838,#38383838,#3F3F    ,#0       ,#0       ,#0       ,#0       ,
#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,#0       ,#0       ,
#0       ,#383C1F0F,#38383838,#3C383838,#F1F     ,#0       ,#0       ,
#0       ,#0       ,#1C3CF8F0,#0       ,#3C1C0000,#F0F8    ,#0       ,
#0       ,#0       ,#0       ,#383C1F0F,#38383838,#3C383838,#F1F     ,
#0       ,#0       ,#1C1C1C1C,#1C1C1C1C,#1C1CFCFC,#1C1C1C1C,#1C1C1C1C,
#FCFC    ,#0       ,#0       ,#0       ,#0       ,#383C1F0F,#3F3F3838,
#3C383838,#F1F     ,#0       ,#0       ,#0       ,#0       ,#1C3CF8F0,
#FCFC1C1C,#0       ,#F8F8    ,#0       ,#0       ,#E0F0703 ,#E0E0E0E ,
#E0E3F3F ,#E0E0E0E ,#E0E0E0E ,#E0E     ,#0       ,#0       ,#3838F0E0,
#0       ,#E0E0    ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#383C1F0F,#38383838,#3C383838,#F1F     ,#0       ,
#1F1F    ,#0       ,#0       ,#1C1CFCFC,#1C1C1C1C,#1C1C1C1C,#1C1CFCFC,
#3C1C1C1C,#F0F8    ,#38383838,#38383838,#38383F3F,#38383838,#38383838,
#3838    ,#0       ,#0       ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,
#1C1C1C1C,#1C1C    ,#0       ,#0       ,#1010101 ,#0       ,#1010F0F ,
#1010101 ,#1010101 ,#1F1F    ,#0       ,#0       ,#C0C0C0C0,#0       ,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#FCFC    ,#0       ,#0       ,#0       ,
#0       ,#101     ,#0       ,#0       ,#0       ,#1E1C0000,#70F     ,
#38383838,#0       ,#3838F8F8,#38383838,#38383838,#38383838,#78383838,
#E0F0    ,#38383838,#38383838,#38383838,#3F3F3B39,#3838393B,#3838    ,
#0       ,#0       ,#0       ,#0       ,#E070381C,#80C0    ,#70E0C080,
#1C38    ,#0       ,#0       ,#1010F0F ,#1010101 ,#1010101 ,#1010101 ,
#1010101 ,#1F1F    ,#0       ,#0       ,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#FCFC    ,#0       ,#0       ,#0       ,#0       ,
#71737F7F,#71717171,#71717171,#7171    ,#0       ,#0       ,#0       ,
#0       ,#C7CFFE7C,#C7C7C7C7,#C7C7C7C7,#C7C7    ,#0       ,#0       ,
#0       ,#0       ,#38383F3F,#38383838,#38383838,#3838    ,#0       ,
#0       ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,#1C1C1C1C,#1C1C    ,
#0       ,#0       ,#0       ,#0       ,#383C1F0F,#38383838,#3C383838,
#F1F     ,#0       ,#0       ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,
#3C1C1C1C,#F0F8    ,#0       ,#0       ,#0       ,#0       ,#38383F3F,
#38383838,#38383838,#38383F3F,#38383838,#3838    ,#0       ,#0       ,
#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,#0       ,#0       ,
#0       ,#383C1F0F,#38383838,#3C383838,#F1F     ,#0       ,#0       ,
#0       ,#0       ,#1C1CFCFC,#1C1C1C1C,#1C1C1C1C,#1C1CFCFC,#1C1C1C1C,
#1C1C    ,#0       ,#0       ,#1C1C1F1F,#1C1C1C1C,#1C1C1C1C,#1C1C    ,
#0       ,#0       ,#0       ,#0       ,#1C3CF8F0,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#383C1F0F,#1071E3C ,
#3C380000,#F1F     ,#0       ,#0       ,#0       ,#0       ,#1C3CF8F0,
#E0800000,#3C1C3C78,#F0F8    ,#0       ,#0       ,#7070000 ,#7070707 ,
#7073F3F ,#7070707 ,#7070707 ,#103     ,#0       ,#0       ,#0       ,
#0       ,#F0F0    ,#0       ,#9C1C0000,#F0F8    ,#0       ,#0       ,
#0       ,#0       ,#38383838,#38383838,#3C383838,#F1F     ,#0       ,
#0       ,#0       ,#0       ,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#FCFC    ,
#0       ,#0       ,#0       ,#0       ,#1C383838,#E0E1C1C ,#3030707 ,
#101     ,#0       ,#0       ,#0       ,#0       ,#1C0E0E0E,#38381C1C,
#E0E07070,#C0C0    ,#0       ,#0       ,#0       ,#0       ,#70707070,
#71717171,#3F3F7B7B,#C0C     ,#0       ,#0       ,#0       ,#0       ,
#87870707,#C7C7C7C7,#7E7EEFEF,#1818    ,#0       ,#0       ,#0       ,
#0       ,#1C1C3838,#7070E0E ,#1C1C0E0E,#3838    ,#0       ,#0       ,
#0       ,#0       ,#70703838,#C0C0E0E0,#7070E0E0,#3838    ,#0       ,
#0       ,#0       ,#0       ,#38383838,#E0E1C1C ,#3030707 ,#101     ,
```

```
#3030101 ,#707     ,#0       ,#0       ,#E0E0E0E ,#1C1C1C1C,#B8B83838,
#E0E0F0F0,#8080C0C0,#0       ,#0       ,#0       ,#3F3F    ,#3010000 ,
#381C0E07,#3F3F    ,#0       ,#0       ,#0       ,#0       ,#381CFCFC,
#80C0E070,#0       ,#FCFC    ,#0       ,#0       ,#1010000 ,#1010101 ,
#30F0301 ,#1010101 ,#1010101 ,#0       ,#0       ,#0       ,#C0E0F878,
#C0C0C0C0,#800080C0,#C0C0C0C0,#E0C0C0C0,#78F8    ,#0       ,#0       ,
#1010101 ,#1010101 ,#101     ,#1010101 ,#1010101 ,#101     ,#0       ,
#0       ,#C0C0C0C0,#C0C0C0C0,#C0C0    ,#C0C0C0C0,#C0C0C0C0,#C0C0    ,
#0       ,#0       ,#1030F0F ,#1010101 ,#1       ,#1010101 ,#3010101 ,
#F0F     ,#0       ,#0       ,#C0C08000,#C0C0C0C0,#E078E0C0,#C0C0C0C0,
#C0C0C0C0,#80      ,#0       ,#0       ,#1F0F0000,#383839  ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#8E0E0000,#78FCCE  ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#3030101 ,
#E0E0707 ,#38381C1C,#7F7F    ,#0       ,#0       ,#0       ,#0       ,
#80800000,#E0E0C0C0,#38387070,#FCFC    ,#0       ,#0       ,#0       ,
#0       ,#383C1F0F,#38383838,#38383838,#38383838,#3C383838,#1010F1F ,
#707     ,#707     ,#1C3CF8F0,#0       ,#0       ,#0       ,#3C1C0000,
#C0C0F0F8,#E0E0C080,#80C0    ,#1C000000,#1C1C    ,#38383838,#38383838,
#3C383838,#F1F     ,#0       ,#0       ,#38000000,#3838    ,#1C1C1C1C,
#1C1C1C1C,#1C1C1C1C,#FCFC    ,#0       ,#0       ,#1000000 ,#703     ,
#383C1F0F,#3F3F3838,#3C383838,#F1F     ,#0       ,#0       ,#C0E07038,
#80      ,#1C3CF8F0,#FCFC1C1C,#0       ,#F8F8    ,#0       ,#0       ,
#7030100 ,#1C0E    ,#1F1F    ,#1F0F0000,#3C38383C,#F1F     ,#0       ,
#0       ,#70E0C080,#1C38    ,#1C3CF8F0,#FCFC1C1C,#1C1C1C1C,#FCFC    ,
#0       ,#0       ,#1C000000,#1C1C    ,#1F1F    ,#1F0F0000,#3C38383C,
#F1F     ,#0       ,#0       ,#38000000,#3838    ,#1C3CF8F0,#FCFC1C1C,
#1C1C1C1C,#FCFC    ,#0       ,#0       ,#3070E1C ,#1       ,#1F1F    ,
#1F0F0000,#3C38383C,#F1F     ,#0       ,#0       ,#80000000,#E0C0    ,
#1C3CF8F0,#FCFC1C1C,#1C1C1C1C,#FCFC    ,#0       ,#0       ,#6060703 ,
#307     ,#1F1F    ,#1F0F0000,#3C38383C,#F1F     ,#0       ,#0       ,
#6060E0C0,#C0E0    ,#1C3CF8F0,#FCFC1C1C,#1C1C1C1C,#FCFC    ,#0       ,
#0       ,#0       ,#0       ,#383C1F0F,#38383838,#3C383838,#1010F1F ,
#1010000 ,#0       ,#0       ,#0       ,#1C3CF8F0,#0       ,#3C1C0000,
#E0C0F0F8,#C0E07070,#0       ,#7030100 ,#1C0E    ,#383C1F0F,#3F3F3838,
#3C383838,#F1F     ,#0       ,#0       ,#70E0C080,#1C38    ,#1C3CF8F0,
#FCFC1C1C,#0       ,#F8F8    ,#0       ,#0       ,#1C000000,#1C1C    ,
#383C1F0F,#3F3F3838,#3C383838,#F1F     ,#0       ,#0       ,#38000000,
#3838    ,#1C3CF8F0,#FCFC1C1C,#0       ,#F8F8    ,#0       ,#0       ,
#103070E ,#0       ,#383C1F0F,#3F3F3838,#3C383838,#F1F     ,#0       ,
#0       ,#C0800000,#70E0    ,#1C3CF8F0,#FCFC1C1C,#0       ,#F8F8    ,
#0       ,#0       ,#1C000000,#1C1C    ,#1010F0F ,#1010101 ,#1010101 ,
#1F1F    ,#0       ,#0       ,#38000000,#3838    ,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#FCFC    ,#0       ,#0       ,#7030100 ,#1C0E    ,#1010F0F ,
#1010101 ,#1010101 ,#1F1F    ,#0       ,#0       ,#70E0C080,#1C38    ,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#FCFC    ,#0       ,#0       ,#103070E ,
#0       ,#1010F0F ,#1010101 ,#1010101 ,#1F1F    ,#0       ,#0       ,
#C0800000,#70E0    ,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#FCFC    ,#0       ,
#0       ,#1C1C1C00,#7030000 ,#38381C0E,#38383F3F,#38383838,#3838    ,
#0       ,#0       ,#38383800,#E0C00000,#1C1C3870,#1C1CFCFC,#1C1C1C1C,
#1C1C    ,#0       ,#3000000 ,#70E0E07 ,#7030003 ,#38381C0E,#38383F3F,
#38383838,#3838    ,#0       ,#E0000000,#F03838F0,#E0C000E0,#1C1C3870,
#1C1CFCFC,#1C1C1C1C,#1C1C    ,#0       ,#0       ,#E070301 ,#3F3F001C,
#38383838,#38383F3F,#38383838,#3F3F    ,#0       ,#E0000000,#80C0    ,
#FCFC0000,#0       ,#E0E0    ,#0       ,#FCFC    ,#0       ,#0       ,
#0       ,#0       ,#3073E3C ,#3F1F0303,#73737373,#1C3E    ,#0       ,
#0       ,#0       ,#0       ,#9C9CF870,#FCFC9C9C,#CC8C8080,#78FC    ,
#0       ,#0       ,#7070301 ,#E0E0E0E ,#1C1C1C1C,#38383F3F,#70707070,
#7070    ,#0       ,#0       ,#E0E0FEFE,#E0E0E0E0,#FCFCE0E0,#E0E0E0E0,
#E0E0E0E0,#FEFE    ,#0       ,#0       ,#7030100 ,#1C0E    ,#383C1F0F,
#38383838,#3C383838,#F1F     ,#0       ,#0       ,#70E0C080,#1C38    ,
#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,#0       ,#1C000000,
#1C1C    ,#383C1F0F,#38383838,#3C383838,#F1F     ,#0       ,#0       ,
#38000000,#3838    ,#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,
#0       ,#103070E ,#0       ,#383C1F0F,#38383838,#3C383838,#F1F     ,
```

```
#0       ,#0       ,#C0800000,#70E0    ,#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,
#F0F8    ,#0       ,#0       ,#7030100 ,#1C0E    ,#38383838,#38383838,
#3C383838,#F1F     ,#0       ,#0       ,#70E0C080,#1C38    ,#1C1C1C1C,
#1C1C1C1C,#1C1C1C1C,#FCFC    ,#0       ,#0       ,#103070E ,#0       ,
#38383838,#38383838,#3C383838,#F1F     ,#0       ,#0       ,#C0800000,
#70E0    ,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#FCFC    ,#0       ,#0       ,
#1C000000,#1C1C    ,#1C383838,#E0E1C1C ,#3030707 ,#101     ,#3030101 ,
#707     ,#1C000000,#1C1C    ,#E0E0E0E ,#1C1C1C1C,#B8B83838,#E0E0F0F0,
#8080C0C0,#0       ,#1C1C1C  ,#383C1F0F,#38383838,#38383838,#3C383838,
#F1F     ,#0       ,#0       ,#383838  ,#1C3CF8F0,#1C1C1C1C,#1C1C1C1C,
#3C1C1C1C,#F0F8    ,#0       ,#0       ,#1C1C1C  ,#38383838,#38383838,
#38383838,#3C383838,#F1F     ,#0       ,#0       ,#383838  ,#1C1C1C1C,
#1C1C1C1C,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,#0       ,#1010000 ,
#3F1F0101,#71717179,#1F3F7971,#1010101 ,#0       ,#0       ,#0       ,
#C0C00000,#FEFCC0C0,#C0C0C0CE,#FCFECEC0,#C0C0C0C0,#0       ,#0       ,
#0       ,#1010100 ,#1010101 ,#7070101 ,#1010101 ,#3F1F0303,#1F3F7373,
#0       ,#0       ,#CECEFEFC,#C0C0C0C0,#F0F0C0C0,#C0C0C0C0,#80808080,
#1EBFF3C0,#0       ,#0       ,#38387070,#E0E1C1C ,#1030707 ,#1011F1F ,
#1011F1F ,#101     ,#0       ,#0       ,#E0E0707 ,#38381C1C,#C0E07070,
#C0C0FCFC,#C0C0FCFC,#C0C0    ,#0       ,#0       ,#38383F3F,#38383838,
#3F3F3838,#3B383838,#3838383B,#3838    ,#0       ,#0       ,#1C3CF8F0,
#1C1C1C1C,#F0F83C1C,#F8E0E000,#E0E0E0F8,#7CFC    ,#0       ,#0       ,
#0       ,#1010000 ,#F0F0101 ,#3030303 ,#7070303 ,#67670707,#3E7F    ,
#0       ,#E6E6FE7C,#C0C0E0E0,#F8F8C0C0,#80808080,#8080    ,#0       ,
#0       ,#0       ,#3010000 ,#E07     ,#1F1F    ,#1F0F0000,#3C38383C,
#F1F     ,#0       ,#0       ,#80C0E070,#0       ,#1C3CF8F0,#FCFC1C1C,
#1C1C1C1C,#FCFC    ,#0       ,#0       ,#3010000 ,#E07     ,#1010F0F ,
#1010101 ,#1010101 ,#1F1F    ,#0       ,#0       ,#80C0E070,#0       ,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#FCFC    ,#0       ,#0       ,#3010000 ,
#E07     ,#383C1F0F,#38383838,#3C383838,#F1F     ,#0       ,#0       ,
#80C0E070,#0       ,#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,
#0       ,#3010000 ,#E07     ,#38383838,#38383838,#3C383838,#F1F     ,
#0       ,#0       ,#80C0E070,#0       ,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,
#FCFC    ,#0       ,#0       ,#38391F0E,#0       ,#38383F3F,#38383838,
#38383838,#3838    ,#0       ,#0       ,#E0F03838,#0       ,#1C3CF8F0,
#1C1C1C1C,#1C1C1C1C,#1C1C    ,#0       ,#0       ,#38391F0E,#3C380000,
#3F3F3E3E,#39393B3B,#38383838,#3838    ,#0       ,#0       ,#E0F03838,
#1C1C0000,#1C1C1C1C,#DCDC9C9C,#7C7CFCFC,#1C3C    ,#0       ,#0       ,
#0       ,#0       ,#1F1F    ,#1F0F0000,#3C38383C,#F1F     ,#3F3F    ,
#0       ,#0       ,#0       ,#1C3CF8F0,#FCFC1C1C,#1C1C1C1C,#FCFC    ,
#FCFC    ,#0       ,#0       ,#0       ,#383C1F0F,#38383838,#3C383838,
#F1F     ,#3F3F    ,#0       ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,
#3C1C1C1C,#F0F8    ,#FCFC    ,#0       ,#3070703 ,#3030000 ,#7070303 ,
#1C1C0E0E,#3C383838,#F1F     ,#0       ,#0       ,#80C0C080,#80800000,
#8080    ,#0       ,#3C1C0000,#F0F8    ,#0       ,#0       ,#0       ,
#0       ,#38383F3F,#3838    ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#FCFC    ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#3F3F    ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#1C1CFCFC,#1C1C    ,#0       ,
#0       ,#0       ,#0       ,#C3C1C0C ,#C0C0C0C ,#1013E3E ,#7070303 ,
#1C1C0E0E,#3838    ,#0       ,#0       ,#1C1C0E0E,#70703838,#C0C0E0E0,
#606BEBC ,#3030180C,#3E3E    ,#0       ,#0       ,#C3C1C0C ,#C0C0C0C ,
#1013E3E ,#7070303 ,#1C1C0E0E,#3838    ,#0       ,#0       ,#1C1C0E0E,
#70703838,#C0C0E0E0,#36369E8E,#FFFF6666,#606     ,#0       ,#0       ,
#70F0F07 ,#7070000 ,#F070707 ,#F0F0F0F ,#F0F0F0F ,#70F     ,#0       ,
#0       ,#808000  ,#0       ,#80000000,#80808080,#80808080,#80      ,
#0       ,#0       ,#0       ,#7070303 ,#1C1C0E0E,#1C1C3939,#7070E0E ,
#303     ,#0       ,#0       ,#0       ,#38389C9C,#E0E07070,#E0E0C0C0,
#38387070,#9C9C    ,#0       ,#0       ,#0       ,#1C1C3939,#7070E0E ,
#7070303 ,#1C1C0E0E,#3939    ,#0       ,#0       ,#0       ,#E0E0C0C0,
#38387070,#38389C9C,#E0E07070,#C0C0    ,#0       ,#22880000,#22882288,
#22882288,#22882288,#22882288,#22882288,#22882288,#22882288,#22882288,
#22882288,#22882288,#22882288,#22882288,#22882288,#22882288,#22882288,
#AA552288,#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,
```

```
#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,#AA55AA55,
#AA55AA55,#AA55AA55,#DD77AA55,#DD77DD77,#DD77DD77,#DD77DD77,#DD77DD77,
#DD77DD77,#DD77DD77,#DD77DD77,#DD77DD77,#DD77DD77,#DD77DD77,#DD77DD77,
#DD77DD77,#DD77DD77,#DD77DD77,#DD77DD77,#101DD77 ,#1010101 ,#1010101 ,
#1010101 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,#C0C00101,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#101     ,
#1010101 ,#1010101 ,#1010101 ,#101FFFF ,#1010101 ,#1010101 ,#1010101 ,
#C0C00101,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#101C0C0 ,#1010101 ,#1010101 ,#FFFF0101,#FFFF0101,#1010101 ,
#1010101 ,#1010101 ,#C0C00101,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#707C0C0 ,#7070707 ,#7070707 ,#7070707 ,
#707FFFF ,#7070707 ,#7070707 ,#7070707 ,#38380707,#38383838,#38383838,
#38383838,#38383838,#38383838,#38383838,#38383838,#3838    ,#0       ,
#0       ,#0       ,#707FFFF ,#7070707 ,#7070707 ,#7070707 ,#707     ,
#0       ,#0       ,#0       ,#3838F8F8,#38383838,#38383838,#38383838,
#3838    ,#0       ,#0       ,#FFFF0000,#FFFF0101,#1010101 ,#1010101 ,
#1010101 ,#101     ,#0       ,#0       ,#C0C00000,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#707C0C0 ,#7070707 ,#7070707 ,#FFFF0707,#FFFF0000,
#7070707 ,#7070707 ,#7070707 ,#38380707,#38383838,#38383838,#38383838,
#38383838,#38383838,#38383838,#38383838,#7073838 ,#7070707 ,#7070707 ,
#7070707 ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,#38380707,#38383838,
#38383838,#38383838,#38383838,#38383838,#38383838,#38383838,#3838    ,
#0       ,#0       ,#FFFF0000,#FFFF0000,#7070707 ,#7070707 ,#7070707 ,
#707     ,#0       ,#0       ,#F8F80000,#38383838,#38383838,#38383838,
#38383838,#7073838 ,#7070707 ,#7070707 ,#FFFF0707,#FFFF0000,#0       ,
#0       ,#0       ,#38380000,#38383838,#38383838,#38383838,#F8F83838,
#0       ,#0       ,#0       ,#7070000 ,#7070707 ,#7070707 ,#7070707 ,
#FFFF    ,#0       ,#0       ,#0       ,#38380000,#38383838,#38383838,
#38383838,#F8F8    ,#0       ,#0       ,#0       ,#1010000 ,#1010101 ,
#1010101 ,#FFFF0101,#FFFF0101,#0       ,#0       ,#0       ,#C0C00000,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#101FFFF ,#1010101 ,#1010101 ,
#1010101 ,#101     ,#0       ,#0       ,#0       ,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#101C0C0 ,#1010101 ,#1010101 ,#1010101 ,#101     ,
#0       ,#0       ,#0       ,#C0C00000,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,
#FFFF    ,#0       ,#0       ,#0       ,#1010000 ,#1010101 ,#1010101 ,
#1010101 ,#FFFF    ,#0       ,#0       ,#0       ,#C0C00000,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#FFFF    ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#101FFFF ,#1010101 ,#1010101 ,#1010101 ,
#101     ,#0       ,#0       ,#0       ,#C0C0FFFF,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#101C0C0 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,
#1010101 ,#1010101 ,#C0C00101,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0FFFF,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0    ,#0       ,#0       ,#0       ,
#FFFF    ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#FFFF    ,#0       ,#0       ,#0       ,#1010000 ,#1010101 ,
#1010101 ,#1010101 ,#101FFFF ,#1010101 ,#1010101 ,#1010101 ,#C0C00101,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0FFFF,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,
#101C0C0 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,
#1010101 ,#C0C00101,#C0C0C0C0,#C0C0C0C0,#FFFFC0C0,#FFFFC0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0C0C0,#707C0C0 ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,
#7070707 ,#7070707 ,#7070707 ,#38380707,#38383838,#38383838,#38383838,
#38383F3F,#38383838,#38383838,#38383838,#7073838 ,#7070707 ,#7070707 ,
#7070707 ,#7070707 ,#0       ,#0       ,#0       ,#38380000,#38383838,
#38383838,#3F3F3838,#FFFF0000,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#7070000 ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,
#707     ,#0       ,#0       ,#FFFF0000,#3F3F0000,#38383838,#38383838,
#38383838,#7073838 ,#7070707 ,#7070707 ,#FFFF0707,#FFFF0000,#0       ,
#0       ,#0       ,#38380000,#38383838,#38383838,#3F3F3838,#FFFF0000,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#FFFF0000,
#FFFF0000,#7070707 ,#7070707 ,#7070707 ,#707     ,#0       ,#0       ,
#FFFF0000,#3F3F0000,#38383838,#38383838,#38383838,#7073838 ,#7070707 ,
#7070707 ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,#38380707,
#38383838,#38383838,#3F3F3838,#3F3F0000,#38383838,#38383838,#38383838,
#3838    ,#0       ,#0       ,#FFFF0000,#FFFF0000,#0       ,#0       ,
```

```
#0        ,#0        ,#0        ,#0        ,#FFFF0000,#FFFF0000,#0        ,
#0        ,#0        ,#7070000 ,#7070707 ,#7070707 ,#FFFF0707,#FFFF0000,
#7070707 ,#7070707 ,#7070707 ,#38380707,#38383838,#38383838,#3F3F3838,
#3F3F0000,#38383838,#38383838,#38383838,#1013838 ,#1010101 ,#1010101 ,
#FFFF0101,#FFFF0000,#0        ,#0        ,#0        ,#C0C00000,#C0C0C0C0,
#C0C0C0C0,#FFFFC0C0,#FFFF0000,#0        ,#0        ,#0        ,#7070000 ,
#7070707 ,#7070707 ,#7070707 ,#FFFF     ,#0        ,#0        ,#0        ,
#38380000,#38383838,#38383838,#38383838,#FFFF     ,#0        ,#0        ,
#0        ,#0        ,#0        ,#0        ,#FFFF0000,#FFFF0000,#1010101 ,
#1010101 ,#1010101 ,#101      ,#0        ,#0        ,#FFFF0000,#FFFF0000,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0     ,#0        ,#0        ,#0        ,
#707FFFF ,#7070707 ,#7070707 ,#7070707 ,#707      ,#0        ,#0        ,
#0        ,#3838FFFF,#38383838,#38383838,#38383838,#7073838 ,#7070707 ,
#7070707 ,#7070707 ,#707      ,#0        ,#0        ,#0        ,#38380000,
#38383838,#38383838,#38383838,#FFFF     ,#0        ,#0        ,#0        ,
#1010000 ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,#0        ,#0        ,
#0        ,#C0C00000,#C0C0C0C0,#C0C0C0C0,#FFFFC0C0,#FFFFC0C0,#0        ,
#0        ,#0        ,#0        ,#0        ,#0        ,#1010000 ,#1010101 ,
#1010101 ,#1010101 ,#1010101 ,#101      ,#0        ,#0        ,#FFFF0000,
#FFFFC0C0,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#C0C0     ,#0        ,#0        ,
#0        ,#7070707 ,#7070707 ,#7070707 ,#7070707 ,#707      ,#0        ,
#0        ,#0        ,#3838FFFF,#38383838,#38383838,#38383838,#7073838 ,
#7070707 ,#7070707 ,#7070707 ,#707FFFF ,#7070707 ,#7070707 ,#7070707 ,
#38380707,#38383838,#38383838,#38383838,#3838FFFF,#38383838,#38383838,
#38383838,#1013838 ,#1010101 ,#1010101 ,#FFFF0101,#FFFF0101,#1010101 ,
#1010101 ,#1010101 ,#C0C00101,#C0C0C0C0,#C0C0C0C0,#FFFFC0C0,#FFFFC0C0,
#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,#101C0C0 ,#1010101 ,#1010101 ,#1010101 ,
#FFFF     ,#0        ,#0        ,#0        ,#C0C00000,#C0C0C0C0,#C0C0C0C0,
#C0C0C0C0,#C0C0     ,#0        ,#0        ,#0        ,#0        ,#0        ,
#0        ,#0        ,#1010101 ,#1010101 ,#1010101 ,#1010101 ,#101      ,
#0        ,#0        ,#0        ,#C0C0FFFF,#C0C0C0C0,#C0C0C0C0,#C0C0C0C0,
#FFFFC0C0,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,
#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,
#FFFFFFFF,#FFFFFFFF,#FFFF     ,#0        ,#0        ,#0        ,#FFFF0000,
#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFF     ,#0        ,#0        ,#0        ,
#FFFF0000,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,
#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFF     ,#0        ,
#0        ,#0        ,#0        ,#0        ,#0        ,#0        ,#0        ,
#0        ,#0        ,#0        ,#0        ,#0        ,#0        ,#0        ,
#FFFF0000,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,
#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFF     ,#0        ,
#0        ,#0        ,#FFFF0000,#FFFFFFFF,#FFFFFFFF,#FFFFFFFF,#FFFF     ,
#0        ,#0        ,#0        ,#0        ,#0        ,#0        ,#70783F1F,
#70707070,#78707070,#1F3F     ,#0        ,#0        ,#0        ,#0        ,
#78FCCE87,#38383838,#FC783838,#C7EE     ,#0        ,#0        ,#7070707 ,
#E0E07070 ,#E0E0E0E ,#1C1C1C1C,#3F3F1C1C,#38383838,#3838     ,#0        ,
#1C1CF8F0,#70381C1C,#383870E0,#1C1C1C1C,#E0F0783C,#0        ,#0        ,
#0        ,#38383F3F,#38383838,#38383838,#38383838,#38383838,#3838     ,
#0        ,#0        ,#C0CFCFC ,#0        ,#0        ,#0        ,#0        ,
#0        ,#0        ,#0        ,#0        ,#0        ,#6E6E3F1F,#1C1C0E0E,
#38381C1C,#3838     ,#0        ,#0        ,#0        ,#0        ,#3838FEFE,
#70703838,#E0E07070,#E0E0     ,#0        ,#0        ,#1C1C3F3F,#7070E0E ,
#1010303 ,#7070303 ,#1C1C0E0E,#3F3F     ,#0        ,#0        ,#C0CFCFC ,
#0        ,#C0C08080,#8080     ,#C0C0000 ,#FCFC     ,#0        ,#0        ,
#0        ,#0        ,#70783F1F,#70707070,#78707070,#1F3F     ,#0        ,
#0        ,#0        ,#0        ,#3870FFFF,#38383838,#78383838,#E0F0     ,
#0        ,#0        ,#0        ,#0        ,#E0E0E0E ,#E0E0E0E ,#F0E0E0E ,
#E0E0E0F ,#6E0E0E0E,#3C7E     ,#0        ,#0        ,#7070707 ,#7070707 ,
#F0707C7 ,#FCFE     ,#0        ,#0        ,#0        ,#0        ,#1C1C1F0F,
#3930101 ,#7070303 ,#707      ,#0        ,#0        ,#0        ,#0        ,
#C0C0FCFC,#80808080,#8080     ,#0        ,#0        ,#0        ,#1C1C0E0E,
#F0F0101 ,#1C1C1C1C,#F0F1C1C ,#1C1C1C01 ,#707      ,#0        ,#0        ,
#C0C0F0F0,#F0F0C0C0,#1C1C1C1C,#F0F01C1C,#C0C0C0C0,#F0F0     ,#0        ,
#0        ,#1C1C1F0F,#70707070,#78787878,#78787878,#E0F0     ,
```

```
#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,#FCFC1C1C,#1C1C1C1C,#3C1C1C1C,
#F0F8    ,#0       ,#0       ,#383C1F0F,#38383838,#38383838,#38383838,
#E0E1C1C ,#3E3E    ,#0       ,#0       ,#1C3CF8F0,#1C1C1C1C,#1C1C1C1C,
#1C1C1C1C,#70703838,#7C7C    ,#0       ,#0       ,#3030100 ,#101     ,
#383C1F0F,#38383838,#3C383838,#F1F     ,#0       ,#0       ,# C8CF8F0,
#E0E0C0C0,#1C3CF8F0,#1C1C1C1C,#3C1C1C1C,#F0F8    ,#0       ,#0       ,
#0       ,#0       ,#71733F1E,#1E3F7371,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#C7E77E3C,#3C7EE7C7,#0       ,#0       ,
#0       ,#0       ,#0       ,#F070000 ,#1C1C1C1C,#1D1D1D1D,#303070F ,
#303     ,#0       ,#0       ,#60600000,#F8F06060,#DCDCDCDC,#9C9C9C9C,
#F0F8    ,#0       ,#0       ,#0       ,#E0F0703 ,#38381C1C,#3F3F3838,
#38383838,#F0E1C1C ,#307     ,#0       ,#0       ,#FCFC    ,#0       ,
#FCFC0000,#0       ,#0       ,#FCFC    ,#0       ,#0       ,#383C1F0F,
#38383838,#38383838,#38383838,#38383838,#3838    ,#0       ,#0       ,
#1C3CF8F0,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#1C1C1C1C,#1C1C    ,#0       ,
#0       ,#0       ,#0       ,#3F3F    ,#3F3F    ,#3F3F    ,#0       ,
#0       ,#0       ,#0       ,#0       ,#FCFC    ,#FCFC    ,#FCFC    ,
#0       ,#0       ,#0       ,#0       ,#3030000 ,#3F3F0303,#3030303 ,
#3F3F0000,#0       ,#0       ,#0       ,#0       ,#80800000,#F8F88080,
#80808080,#F8F80000,#0       ,#0       ,#0       ,#0       ,#3030707 ,
#101     ,#0       ,#3030101 ,#707     ,#1F1F    ,#0       ,#0       ,
#80800000,#E0E0C0C0,#E0E07070,#8080C0C0,#0       ,#F0F0    ,#0       ,
#0       ,#1010000 ,#7070303 ,#7070E0E ,#1010303 ,#0       ,#F0F     ,
#0       ,#0       ,#C0C0E0E0,#8080    ,#0       ,#C0C08080,#E0E0    ,
#F0F0    ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#1010000 ,#1010101 ,#101     ,#0       ,#E0E6FE7C,#E0E0E0E0,#E0E0E0E0,
#E0E0E0E0,#C0C0E0E0,#C0C0C0C0,#101C0C0 ,#1010101 ,#3030101 ,#3030303 ,
#3030303 ,#3030303 ,#1F3F3303,#0       ,#C0C00000,#C0C0C0C0,#8080C0C0,
#80808080,#80808080,#80808080,#808080  ,#0       ,#0       ,#0       ,
#3070703 ,#3F3F0000,#7030000 ,#307     ,#0       ,#0       ,#0       ,
#0       ,#80C0C080,#F8F80000,#C0800000,#80C0    ,#0       ,#0       ,
#0       ,#0       ,#1F0F0000,#3839    ,#38391F0F,#0       ,#0       ,
#0       ,#0       ,#0       ,#9C1C0000,#F0F8    ,#F0F89C1C,#0       ,
#0       ,#0       ,#0       ,#E0E0703 ,#3070E0E ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#3838F0E0,#E0F03838,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#7070300 ,#30707   ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#E0E0C000,
#C0E0E0  ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#3000000 ,#7070707 ,#3       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#C0000000,#E0E0E0E0,#C0      ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#1C1C7878,#7070E0E ,
#1010303 ,#0       ,#7F7F0000,#70707070,#70707070,#70707070,#70707070,
#70707070,#F0F0F0F0,#7070F0F0,#0       ,#6060707 ,#6060606 ,#606     ,
#0       ,#0       ,#0       ,#0       ,#0       ,#6060C080,#60606060,
#6060    ,#0       ,#0       ,#0       ,#0       ,#0       ,#707     ,
#6060301 ,#707     ,#0       ,#0       ,#0       ,#0       ,#0       ,
#6060E0C0,#80C0    ,#E0E0    ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#7070000 ,#7070707 ,#7070707 ,#0       ,#0       ,
#0       ,#0       ,#0       ,#E0E00000,#E0E0E0E0,#E0E0E0E0,#0       ,
#0       ,#0       ,#0       ,#383C1F0F,#77777370,#77777777,#77777777,
#3C387073,#F1F     ,#0       ,#0       ,#E1EFCF8 ,#37F7E707,#7070737 ,
#F7373707,#1E0E07E7,#F8FC    ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,
#0       ,#0       ,#0       ,#0       ,#0       ,#0       ,#1A1A0000]:
```

6.2.3. Make files

```
LIBRARIAN=ilibr
OCCAM=occam
LINK=ilink
CONFIG=iconf
ADDBOOT=iboot
LIBOPT=
OCCOPT=/a
LINKOPT=
CONFOPT=
BOOTOPT=

seeker.btl: seeker.pgm controll.t8h guidance.t8h xbar.t2h gtsei.t2h \
          backgrou.t8h  targetle.t8h  target.t8h  spcontro.t8h  sp.t4h
firstbuf.t8h \
        secondbu.t8h formatte.t8h imagedis.c8h trackdis.c8h b409stub.c2h
\
    hostseek.c8h
      $(CONFIG) seeker /o seeker.btl $(CONFOPT)

HostSeek.c8h:     HostSeek.l8h HostSeek.t8h
      $(LINK) /f HostSeek.l8h /o HostSeek.c8h $(LINKOPT)

HostSeek.t8h:     HostSeek.occ s_header.inc c:\itools\libs\hostio.inc \
        c:\itools\libs\hostio.lib c:\itools\libs\hostio.liu \
        c:\itools\libs\convert.lib gif.c8h loader.c8h runseekr.c8h
      $(OCCAM) HostSeek /t8 /h /o HostSeek.t8h $(OCCOPT)

gif.c8h:    gif.l8h gif.t8h
      $(LINK) /f gif.l8h /o gif.c8h $(LINKOPT)

gif.t8h:    gif.occ  c:\itools\libs\hostio.inc  c:\itools\libs\hostio.lib
\
        c:\itools\libs\hostio.liu c:\itools\libs\convert.lib
      $(OCCAM) gif /t8 /h /o gif.t8h $(OCCOPT)

loader.c8h: loader.l8h loader.t8h
      $(LINK) /f loader.l8h /o loader.c8h $(LINKOPT)

loader.t8h: loader.occ s_header.inc c:\itools\libs\hostio.inc \
        c:\itools\libs\hostio.lib c:\itools\libs\hostio.liu \
        c:\itools\libs\convert.lib
      $(OCCAM) loader /t8 /h /o loader.t8h $(OCCOPT)

runseekr.c8h:     runseekr.l8h runseekr.t8h
      $(LINK) /f runseekr.l8h /o runseekr.c8h $(LINKOPT)

runseekr.t8h:     runseekr.occ s_header.inc c:\itools\libs\hostio.inc \
        c:\itools\libs\hostio.lib c:\itools\libs\hostio.liu \
        c:\itools\libs\convert.lib
      $(OCCAM) runseekr /t8 /h /o runseekr.t8h $(OCCOPT)

controll.t8h:     controll.occ s_header.inc
      $(OCCAM) controll /t8 /h /o controll.t8h $(OCCOPT)

guidance.t8h:     guidance.occ s_header.inc
      $(OCCAM) guidance /t8 /h /o guidance.t8h $(OCCOPT)

xbar.t2h:   xbar.occ s_header.inc
      $(OCCAM) xbar /t2 /h /o xbar.t2h $(OCCOPT)

gtsei.t2h:  gtsei.occ s_header.inc
```

```
      $(OCCAM) gtsei /t2 /h /o gtsei.t2h $(OCCOPT)

backgrou.t8h:     backgrou.occ s_header.inc
      $(OCCAM) backgrou /t8 /h /o backgrou.t8h $(OCCOPT)

targetle.t8h:     targetle.occ s_header.inc
      $(OCCAM) targetle /t8 /h /o targetle.t8h $(OCCOPT)

target.t8h: target.occ s_header.inc
      $(OCCAM) target /t8 /h /o target.t8h $(OCCOPT)

spcontro.t8h:     spcontro.occ s_header.inc
      $(OCCAM) spcontro /t8 /h /o spcontro.t8h $(OCCOPT)

sp.t4h:     sp.occ s_header.inc
      $(OCCAM) sp /t4 /h /o sp.t4h $(OCCOPT)

firstbuf.t8h:     firstbuf.occ
      $(OCCAM) firstbuf /t8 /h /o firstbuf.t8h $(OCCOPT)

secondbu.t8h:     secondbu.occ
      $(OCCAM) secondbu /t8 /h /o secondbu.t8h $(OCCOPT)

formatte.t8h:     formatte.occ
      $(OCCAM) formatte /t8 /h /o formatte.t8h $(OCCOPT)

imagedis.c8h:     imagedis.l8h imagedis.t8h
      $(LINK) /f imagedis.l8h /o imagedis.c8h $(LINKOPT)

imagedis.t8h:     imagedis.occ          crtc.inc          g_header.inc
c:\itools\libs\convert.lib \
        graphics.lib s_header.inc
      $(OCCAM) imagedis /t8 /h /o imagedis.t8h $(OCCOPT)

graphics.lib:     graphics.lbb    b409.t2h    g_line.t8h    g_system.t8h
g_text.t8h
      $(LIBRARIAN) /f graphics.lbb  /o graphics.lib $(LIBOPT)

b409.t2h:   b409.occ crtc.inc
      $(OCCAM) b409 /t2 /h /o b409.t2h $(OCCOPT)

g_line.t8h: g_line.occ g_header.inc
      $(OCCAM) g_line /t8 /h /o g_line.t8h $(OCCOPT)

g_system.t8h:     g_system.occ crtc.inc g_header.inc
      $(OCCAM) g_system /t8 /h /o g_system.t8h $(OCCOPT)

g_text.t8h: g_text.occ g_header.inc
      $(OCCAM) g_text /t8 /h /o g_text.t8h $(OCCOPT)

trackdis.c8h:     trackdis.l8h trackdis.t8h
      $(LINK) /f trackdis.l8h /o trackdis.c8h $(LINKOPT)

trackdis.t8h:     trackdis.occ s_header.inc g_header.inc crtc.inc \
        c:\itools\libs\convert.lib graphics.lib sys6.inc
      $(OCCAM) trackdis /t8 /h /o trackdis.t8h $(OCCOPT)

b409stub.c2h:     b409stub.l2h b409stub.t2h
      $(LINK) /f b409stub.l2h /o b409stub.c2h $(LINKOPT)

b409stub.t2h:     b409stub.occ crtc.inc graphics.lib
      $(OCCAM) b409stub /t2 /h /o b409stub.t2h $(OCCOPT)
```

```
LIBRARIAN=ilibr
OCCAM=occam
LINK=ilink
CONFIG=iconf
ADDBOOT=iboot
LIBOPT=
OCCOPT=
LINKOPT=
CONFOPT=
BOOTOPT=

hostseek.c8h:     hostseek.l8h hostseek.t8h
      $(LINK) /f hostseek.l8h /o hostseek.c8h $(LINKOPT)

hostseek.t8h:     hostseek.occ s_header.inc c:\itools\libs\hostio.inc \
        c:\itools\libs\hostio.lib c:\itools\libs\hostio.liu \
        c:\itools\libs\convert.lib
      $(OCCAM) hostseek /t8 /h /o hostseek.t8h $(OCCOPT)
```

```
LIBRARIAN=ilibr
OCCAM=occam
LINK=ilink
CONFIG=iconf
ADDBOOT=iboot
LIBOPT=
OCCOPT=
LINKOPT=
CONFOPT=
BOOTOPT=

graphics.lib:     graphics.lbb    b409.t2h    g_line.t8h    g_system.t8h
g_text.t8h
      $(LIBRARIAN) /f graphics.lbb  /o graphics.lib $(LIBOPT)

b409.t2h:   b409.occ crtc.inc
      $(OCCAM) b409 /t2 /h /o b409.t2h $(OCCOPT)

g_line.t8h: g_line.occ g_header.inc
      $(OCCAM) g_line /t8 /h /o g_line.t8h $(OCCOPT)

g_system.t8h:     g_system.occ crtc.inc g_header.inc
      $(OCCAM) g_system /t8 /h /o g_system.t8h $(OCCOPT)

g_text.t8h: g_text.occ g_header.inc
      $(OCCAM) g_text /t8 /h /o g_text.t8h $(OCCOPT)
```

```
LIBRARIAN=ilibr
OCCAM=occam
LINK=ilink
CONFIG=iconf
ADDBOOT=iboot
LIBOPT=
OCCOPT=
LINKOPT=
CONFOPT=
BOOTOPT=

test.b8h:   test.c8h
      $(ADDBOOT) test.c8h /o test.b8h $(BOOTOPT)

test.c8h:   test.l8h test.t8h
      $(LINK) /f test.l8h /o test.c8h $(LINKOPT)

test.t8h:   test.occ c:\itools\libs\hostio.inc c:\itools\libs\hostio.lib
\
        c:\itools\libs\hostio.liu c:\itools\libs\convert.lib
      $(OCCAM) test /t8 /h /o test.t8h $(OCCOPT)
```

6.2.4. Link command files

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "gif.l8h"
--
gif.t8h
hostio.lib
convert.lib
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "gif02.18h"
--
gif02.t8h
hostio.lib
convert.lib
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "b409stub.l2h"
--
b409stub.t2h
graphics.lib
occam2h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "HostSeek.l8h"
--
HostSeek.t8h
=gif02.t8h
=loader.t8h
=runseekr.t8h
hostio.lib
convert.lib
gif02.c8h
loader.c8h
runseekr.c8h
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "test.l8h"
--
test.t8h
hostio.lib
convert.lib
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "trackdis.18h"
--
trackdis.t8h
convert.lib
graphics.lib
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "imagedis.18h"
--
imagedis.t8h
convert.lib
graphics.lib
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "loader.18h"
--
loader.t8h
hostio.lib
convert.lib
occam8h.lib
```

```
--
-- Version 2.80, 28th April 1989
--
-- linker command file "runseekr.l8h"
--
runseekr.t8h
hostio.lib
convert.lib
occam8h.lib
```

6.2.5. Motherboard C004 Configuration

```
b0
;  Connect to VAX
  s0 3  t  e1 .

;  Foreground Connections
b1
  c23  t  e0 .
  c22  t  e4 .
b2
  c23  t  e0 .
  c22  t  e4 .
b3
  c23  t  e0 .
  c22  t  e4 .
b4
  c23  t  e0 .
  c22  t  e4 .
b5
  c23  t  e0 .
  c22  t  e4 .
b6
  c23  t  e0 .
  c22  t  e4 .
b7
  c23  t  e0 .
  c22  t  e4 .
b8
  c23  t  e0 .
  c22  t  e4 .
b9
  c23  t  e0 .
  c22  t  e4 .
b10
  c23  t  e0 .
  c22  t  e4 .
b11
  c23  t  e0 .
  c22  t  e4 .
b12
  c23  t  e0 .
  c22  t  e4 .
b13
  c23  t  e0 .
  c22  t  e4 .
b14
  c23  t  e0 .
  c22  t  e4 .
b15
  c23  t  e0 .
  c22  t  e4 .
b16
  c23  t  e0 .
  c22  t  e4 .

b17
;  Graphics B408 and B409
  s1 0  t  e2 .
  s6 0  t  s1 3 .

b18
;  Graphics 2 B410
```

```
  s1 0  t  s15 3 .
  s1 3  t  e0 .

b19
;  Second Graphics Buffer
  c23  t  e0 .
  s1 3  t  e1 .
  s2 3  t  e6 .
  s3 3  t  e7 .
  s4 3  t  e8 .
  s5 3  t  e9 .
  s6 3  t  e12 .
  s7 3  t  e15 .

  s8 3  t  e27 .
  s9 3  t  e26 .
  s10 3  t  e25 .
  s11 3  t  e24 .
  s12 3  t  e23 .
  s13 3  t  e22 .
  s14 3  t  e19 .
  s15 3  t  e16 .

  c22  t  e2 .
  s1 0  t  e3 .
  s2 0  t  e4 .
  s3 0  t  e5 .
  s4 0  t  e10 .
  s5 0  t  e11 .
  s6 0  t  e13 .
  s7 0  t  e14 .

  s8 0  t  e31 .
  s9 0  t  e30 .
  s10 0  t  e29 .
  s11 0  t  e28 .
  s12 0  t  e21 .
  s13 0  t  e20 .
  s14 0  t  e18 .
  s15 0  t  e17 .

b20
;  Signal Processing
  c23  t  e0 .
  s1 3  t  e1 .
  s2 3  t  e6 .
  s3 3  t  e7 .
  s4 3  t  e8 .
  s5 3  t  e9 .
  s6 3  t  e12 .
  s7 3  t  e15 .
  s8 3  t  e16 .
  s9 3  t  e19 .
  s10 3  t  e22 .
  s11 3  t  e23 .
  s12 3  t  e24 .
  s13 3  t  e25 .
  s14 3  t  e26 .
  s15 3  t  e27 .

  c22  t  e2 .
  s1 0  t  e3 .
  s2 0  t  e4 .
  s3 0  t  e5 .
```

```
  s4 0  t  e10 .
  s5 0  t  e11 .
  s6 0  t  e13 .
  s7 0  t  e14 .
  s8 0  t  e17 .
  s9 0  t  e18 .
  s10 0  t  e20 .
  s11 0  t  e21 .
  s12 0  t  e28 .
  s13 0  t  e29 .
  s14 0  t  e30 .
  s15 0  t  e31 .

b21
;  First Graphics Buffer
  c23  t  e0 .
  s1 3  t  e1 .
  s2 3  t  e6 .
  s3 3  t  e7 .
  s4 3  t  e8 .
  s5 3  t  e9 .
  s6 3  t  e12 .
  s7 3  t  e15 .

  s8 3  t  e27 .
  s9 3  t  e26 .
  s10 3  t  e25 .
  s11 3  t  e24 .
  s12 3  t  e23 .
  s13 3  t  e22 .
  s14 3  t  e19 .
  s15 3  t  e16 .

  c22  t  e2 .
  s1 0  t  e3 .
  s2 0  t  e4 .
  s3 0  t  e5 .
  s4 0  t  e10 .
  s5 0  t  e11 .
  s6 0  t  e13 .
  s7 0  t  e14 .

  s8 0  t  e31 .
  s9 0  t  e30 .
  s10 0  t  e29 .
  s11 0  t  e28 .
  s12 0  t  e21 .
  s13 0  t  e20 .
  s14 0  t  e18 .
  s15 0  t  e17 .

b22
;  Second Background Set
  c23  t  e16 .
  s1 3  t  e19 .
  s2 3  t  e22 .
  s3 3  t  e23 .
  s8 3  t  e24 .
  s9 3  t  e25 .
  s10 3  t  e26 .
  s11 3  t  e27 .

  c22  t  e17 .
  s1 0  t  e18
```

```
  s2 0  t  e20 .
  s3 0  t  e21 .
  s8 0  t  e28 .
  s9 0  t  e29 .
  s10 0  t  e30 .
  s11 0  t  e31 .

b23
;  First Background Set
  c23  t  e0 .
  s1 3  t  e1 .
  s2 3  t  e6 .
  s3 3  t  e7 .
  s8 3  t  e8 .
  s9 3  t  e9 .
  s10 3  t  e12 .
  s11 3  t  e15 .

  c22  t  e2 .
  s1 0  t  e3 .
  s2 0  t  e4 .
  s3 0  t  e5 .
  s8 0  t  e10 .
  s9 0  t  e11 .
  s10 0  t  e13 .
  s11 0  t  e14 .

b24
;  Controller
  c22   t  e2 .
  c23   t  e0 .
  s1 0  t  e3 .
  s1 3  t  e1 .
  s2 0  t  e4 .
```